VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

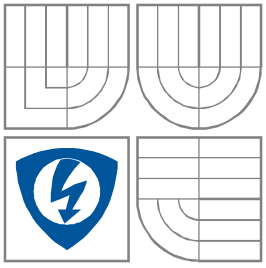
KLASIFIKACE EKG SIGNÁLŮ S POUŽITÍM NEURONOVÝCH SÍTÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE David Loviška
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

KLASIFIKACE EKG SIGNÁLŮ S POUŽITÍM NEURONOVÝCH SÍTÍ

CLASSIFICATION OF ECG BY ARTIFICIAL NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — David Loviška
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Hrubeš
SUPERVISOR

BRNO, 2008

Anotace

Cílem projektu Klasifikace EKG signálů pomocí neuronových sítí je zjednodušit a urychlit práci lékaře. Toho lze dosáhnout vytvořením programu schopného jednoduše a téměř okamžitě klasifikovat EKG signál s použitím umělé neuronové sítě. Vytvořený program poskytne lékaři základní informace o vloženém elektrokardiogramu, jako jsou časové intervaly a amplitudy signálu v jednotlivých zkoumaných úsecích. Následně lékaře upozorní na odchylky od normálu. Součástí programu je i grafické okno se zobrazeným signálem a na něm barevně zvýrazněny body a úseky vyhodnocené programem za zvláštní. V další fázi bude program sám klasifikovat získané údaje a určí nezávisle na lékaři diagnózu, kterou může lékař vyhodnotit a případně vlastním podpisem uznat za skutečnou diagnózu pacienta. Tento program je rovněž vhodný pro několikahodinové, až týdenní záznamy Holterova monitorování EKG.

Abstract

The aim of project with name Classification ECG by artificial neural networks is simplify and speed up working a doctor. That reaches created program that the is capable simply and almost at once classify EKG signal using artificial neuronal nets. Created program will give to the doctor basic information about used electrocardiogram, as are time period and amplitude signal in single surveyed sections. Subsequently will program warn doctor about abnormalities from normal. Part of program is also graphic window with painted signal and on him in color points and partitions marked by program behind special. In next phase program alone classifies gained data and designating without doctor diagnose that doctor can evaluate and in case agreeable it sign and place for true diagnose patient. This program is also fit for data reading from bigger of the number of hours as far as days. It is concerned primarily Holter ECG monitoring.

Bibliografická citace dle ČSN IS0 690

LOVIŠKA, D. *Klasifikace EKG signálů s použitím neuronových sítí ,*
Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 46 s. Vedoucí bakalářské práce Ing. Jan Hrubeš.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Klasifikace EKG signálů s použitím neuronových sítí jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 6. června 2008 .....

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Hrubešovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 6. června 2008 .....

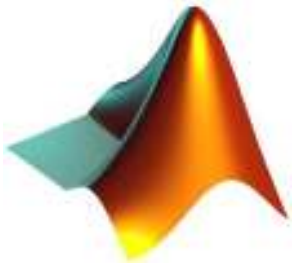
podpis autora

Obsah :

Seznam obrázků :

1 Úvod :

Jeden z mnoha citátů začíná slovy : „Proměnili jsme lékaře v bohy a jejich božství uctíváme tím,že jim svěřujeme svá těla a duše...“ a právě to se dnes plno odborníků snaží změnit a předává nelehkou lékařovu volbu, zkušenost a cit pro naložení s lidským životem chladnému stroji. I tento projekt může být ve finální podobě schopen stanovit pacientovi diagnózu, avšak zákony toto nedovolují. Bude tedy využit ke zpřehlednění a urychlení práce lékaře. Všeobecně však absolvent medicíny má zcela odlišné představy od absolventa libovolné technické školy a proto jsou některé velice technicky dokonalé a poloautomatické diagnostické přístroje považovány za prakticky nepoužitelné. To potvrzuje i odborná veřejnost z nemocničního prostředí. Cílem této části projektu je vytvořit ideální, přehledný, rychlý a jednoduchý grafický program pro klasifikaci EKG signálu a jeho rozdělení do jednotlivých tříd. Aby bylo možno toto udělat, bylo nejprve nutné zvolit nejvhodnější programové prostředí pro tvorbu programu. S neuronovými sítěmi a analogovými elektrokardiogramy si dnes poradí mnoho prostředí. Mezi jasně nejlepší a zároveň nejsložitější patří jazyk C, který dokáže velice dobře a efektivně pracovat s libovolnými typy umělých neuronových sítí. Naopak program LabView od firmy *National Instruments*, který jsem objevil ve školních laboratořích Lékařské diagnostické techniky, dokáže velice jednoduše zpracovávat vložený signál a je graficky velice šťastně zpracován. Nabízí jednoduchou a přehlednou tvorbu okna programu, které je přímo propojené s daty a jednotlivými bloky filtrů a klasifikátorů. Za zlatou střední cestu lze považovat program Matlab od společnost The MathWorks. Jedná se o původně ryze matematickou aplikaci, která se do dnešní doby rozrostla o grafické prostředí tvorby programu a velké množství nástrojů usnadňujících výpočty, jako jsou například filtry, transformace, neuronové sítě s nimiž lze nyní pracovat pomocí jednoduchých příkazů. Grafické prostředí však neumožňuje vytvořit dokonale dynamickou a přehlednou aplikaci jeho běžným uživatelem. Budeme se tedy hlavně věnovat ideovému rozvržení pracovní plochy. Aplikace musí být také schopna zpracovat data získaná z různých přístrojů a Holterů sloužících pro snímání elektrických impulsů lidského těla. K tomu se nabízí nepřeberné množství metod pro úpravu a filtraci. Musíme tedy vyzkoušet různé metody a zvolit tu nejvhodnější pro zpracování obecných neznámých signálů. Jakmile nám aplikace zpracuje signál, tak je nutné v něm vyhledat konkrétní parametry běžně hledané lékařem při vyšetření a tyto parametry přehledně vypsat a zdůraznit v nich odchylky od průměrných hodnot. Ze získaných parametrů lze následně spustit základní klasifikaci signálu. Celý tento soubor procesů by měl být pro lékaře spuštěn stiskem jedné klávesy.



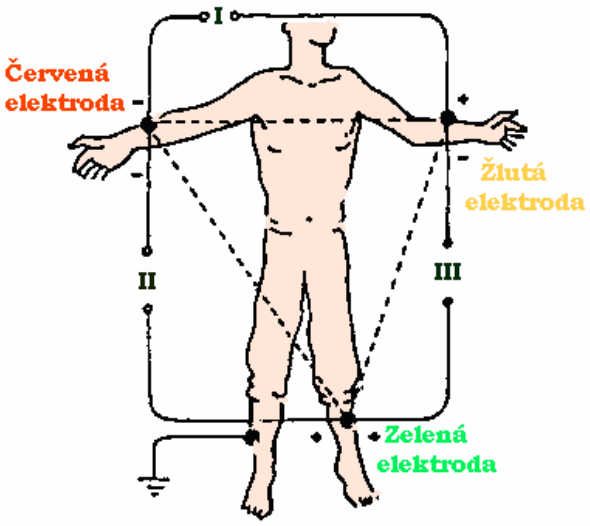
Obr.1
Logo The MathWorks MatLab

1.1 Historie :

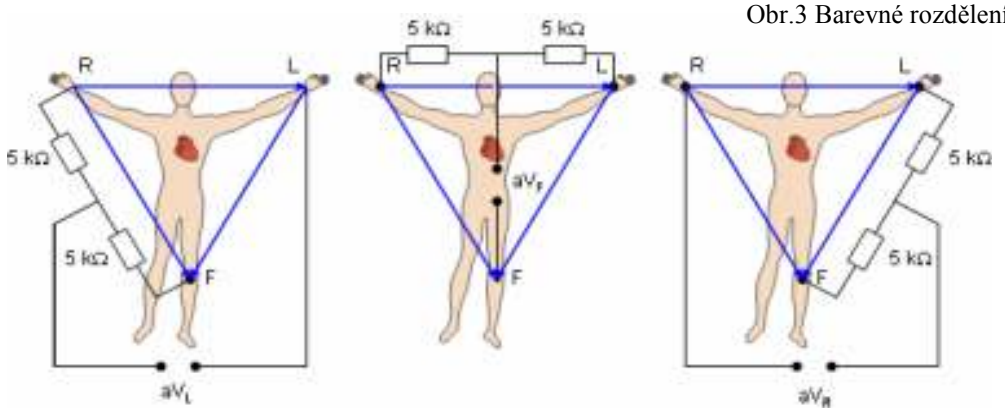
Obr.2
Moderní Holter-EKG monitor

Tomuto oboru byl zrodem objev samovolného vzniku elektrických impulsů u člověka. První zaznamenaný elektrokardiogram (obraz srdečního impulsu) byl zaznamenán v roce 1870 Alexandrem Muirheadem pomocí Thomsonova sifónového zapisovače. Již v roce 1878 bylo popsáno rozdělení srdečních impulsů do dvou fází dnes zvaných QRS (depolarizace komor) a T (repolarizace komor). O 2 roky později byl vylepšen galvanometr a byl použit pro zobrazení přesných srdečních impulsů.Srdeční impulsy se daly pozorovat i zaznamenat pomocí kapilárního elektrometru. Veliký posun v kardiologii s sebou přinesl počátkem 20.století Willem Einthoven, který nejen upravil a vylepšil galvanometr k vytvoření citlivých elektrokardiogramů, ale také popsal rovnostranný trojúhelník s označením vrcholů I, II a III, později nazývaný Einthovenův trojúhelník. Jeho studie dokonce poprvé obsahovala zkratku „ECG“, používanou pro danou specializaci dodnes. Willem dostal za své přínosy později Nobelovu cenu. Následně byla tendence sestrojit přístroj co nejmenší váhy a velikosti, což se roku 1949 podařilo N.J. Holterovi. Následně se podle něj pojmenoval přístroj, který je ve zmodernizované podobě vidět na obrázku č. 2. Roku 1963 se pánům G.M. Baulemu a R. McFeemu podařilo zaznamenat vlastní metodou pomocí lektromagnetického pole člověka první magnetokardiogram , což byl převrat, protože k jeho zjištění nebylo potřebné na člověka připojovat žádné elektrody. Tato metoda však zatím není využívána pro praktickou diagnostiku.

V současné době se používá 12ti svodové EKG, které bylo popsáno již v roce *1942*. Je složeno ze tří bipolárních končetinových svodů, viz. obrázek 3., popsaných jako I. , II. a III. , dále ze tří unipolárních svodů popsaných jako aV_R , aV_L a aV_F , které jsou zobrazeny na obrázku 4.
Zbývajících šest svodů je popsáno jako hrudní svody a jsou označeny V1-V6. Tyto svody se dají najít na spodních dvou žebrech pod srdcem. Signál z těchto svodů se získá srovnáním napětí elektrod vůči středu srdce. Ten se získá propojením všech tří končetinových svodů do hvězdy přes odpor 5kΩ.



Obr.3 Barevné rozdělení svodů



Obr.4 Rozmístění a zapojení unipolárních svodů

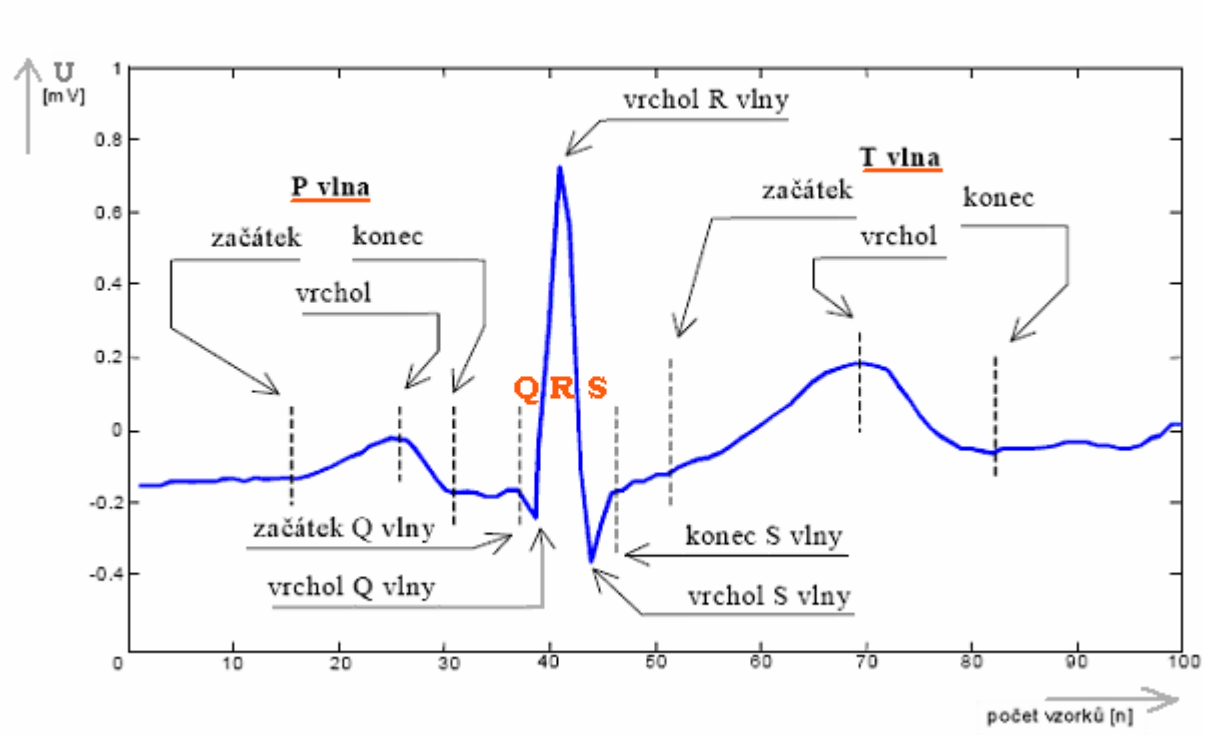
1.2 Pojmy důležité pro stanovení diagnózy :

1.2.1 Sekundární popis periody :

Označení jednotlivých segmentů impulsu :

P vlna–depolarizace síní, 0 –3 mV, do 100 ms;
P-Q segment–izoelektrický, 120 až 200 ms;
Q –první negativní kmit, 0 –25% R vlny, do 30 ms;
R –pozitivní kmit, několik mV, do 100 ms;
S –druhý negativní kmit, 0 až 0.8 mV, do 50 ms;
QRS–depolarizace komor;
S-T segment–izoelektrický interval, pokles nebo vzrůst do 0.1 mV;
Q-T segment–elektrická systola, kolem 400 ms;
T vlna–repolarizace komor, do 8.8 mV, od 100 do 250ms;

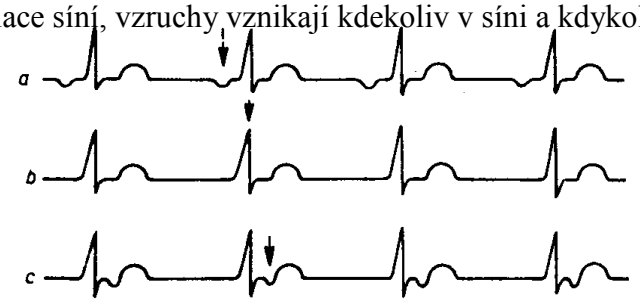
Rozsegmentovaná perioda srdečního impulsu zdravého člověka :



Obr.5 Rozdělení rytmu

1.2.2 Primární popis periody :

Rytmus - sinusový, charakterizovaný nálezem vlny P před celkem QRS
- Junkční (nodální), rozhoduje tvar vlny P podle obrázku 6
- fibrilace síní, vzruchy vznikají kdekoliv v síni a kdykoliv
-



Obr.6
Různé typy nodálního rytmu: a) horní, b) střední, c) dolní.
Šipka ukazuje vlnu P, která je u b) skryta v komorovém komplexu

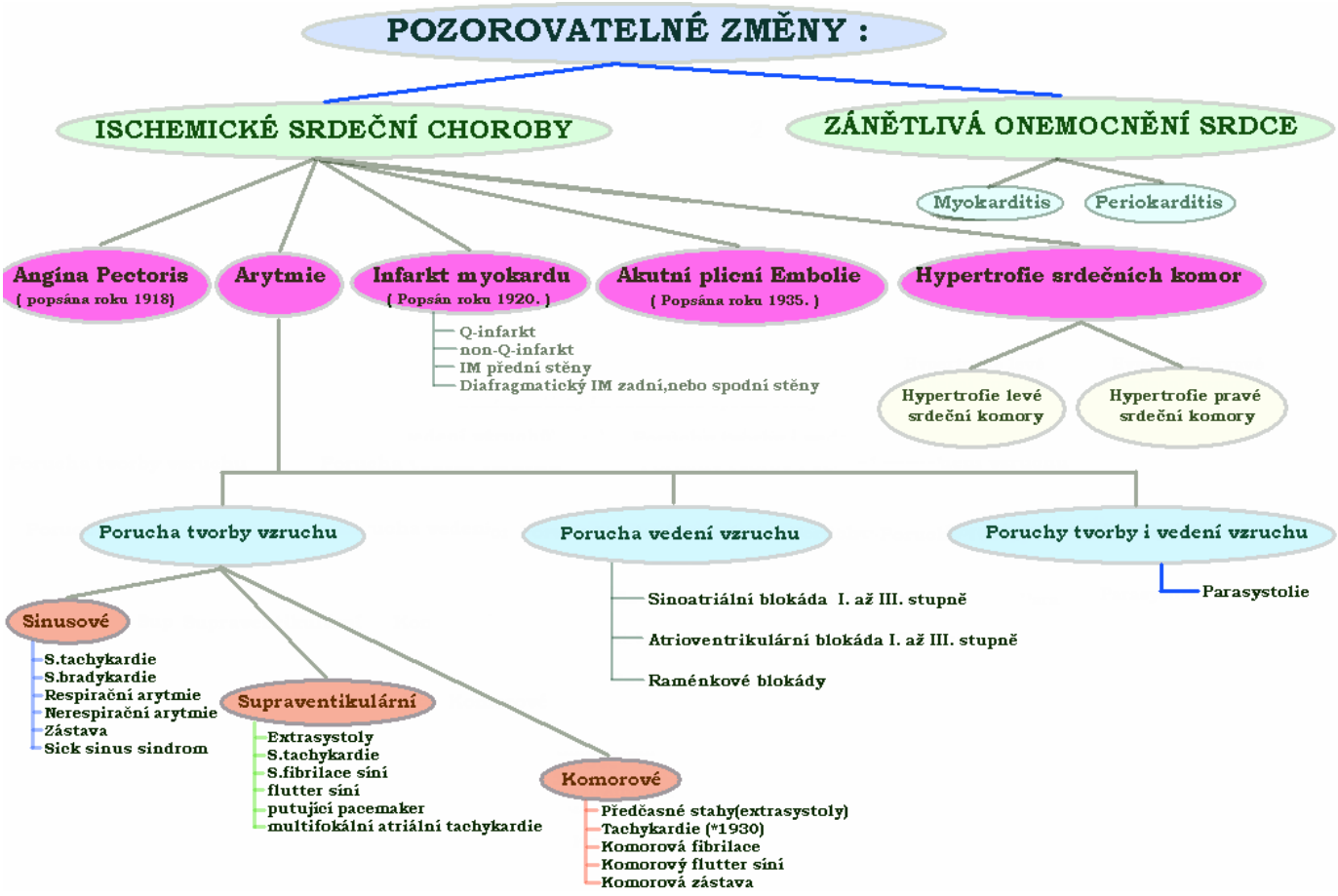
Srdeční akce :

- pravidelná - QRS je od sebe stále stejně daleko
- nepravidelná - QRS je od sebe různě daleko

Frekvence :

- Normální 60-90x QRS za minutu
- Tachykardie 90 a více x QRS za minutu
- Bradykardie 0-60x QRS za minutu

1.3 Diagram vyhotovený za účelem zpřehlednění klasifikace :



Obr.7 Klasifikační strom - diagram

1.4 Metody klasifikace :

1.4.1 Manuální vyšetření :

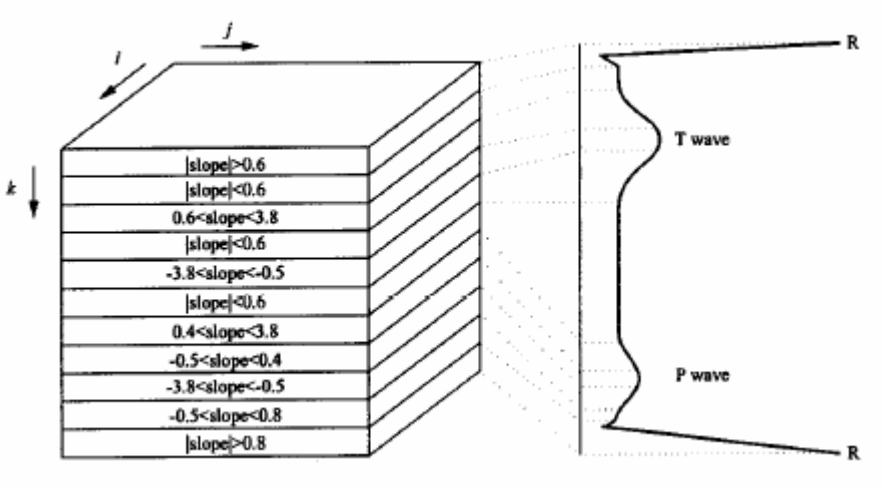
EKG vyšetření je dnes součástí téměř každého diagnostického vyšetření. Je líčovým aspektem při stanovení diagnózy pacienta. Je tedy při nejmenším důležité rozpoznat na získaném elektrokardiogramu veškeré výchylky od normálu a stanovit tím chorobu, či vadu srdečního systému. Nyní je stále běžné, že má rozpoznání odchylek křivek na starost pouze doktor, který buď poznal výchylky ze zkušeností, nebo srovnal získané hodnoty s hodnotami databázovými. Toto rozhodnutí o diagnóze hlavně v případě využití pouze hlavy lékaře nemuselo vést

vždy správným směrem. Po shlédnutí užitečného katalogu, sestaveného kolektivem MUDr. Lenky Borské,Ph.D. , jsem se přesvědčil o nejednoduchosti určení správné diagnózy i s pomocí zmíněného katalogu. Na každé křivce je nutno popsat rytmus, srdeční akce, frekvenci, sklon elektrické srdeční osy a správně analyzovat jednotlivé vlny a kmity a teprve poté lze provést nějaké srovnání s obsáhlým katalogem a následně stanovit diagnózu. Obě tyto manuální metody jsou velice náročné na přemýšlení doktora a záleží pouze na figuře doktora, jakou má úspěšnost ve stanovování diagnózy.

1.4.2 Poloautomatické vyšetření :

S příchodem výpočetní techniky se však naskytla možnost provést alespoň poloautomatické srovnání signálu pacienta s EKG křivkami počítačové databáze, nebo softwarově nastavené vyhledání v signálu znaků fibrilací, či arytmií. K tomuto srovnání se používá zdigitalizovaný signál pacienta. Ten bývá filtrovaný a k vyhodnocení se používají vybrané části tohoto signálu. Metod použitelných pro stanovení poloautomatické diagnózy je více. I systémů, kterými se dají metody realizovat. V literatuře jsem se dočetl o systému WEKA (Waikato Environment for Knowledge Analysis), který pochází z Nového Zélandu. Tento systém je napsán v jazyce Java a anglické vyjádření jeho funkce „Machine Learning Algorithms in Java“ již oznamuje, že se jedná o strojové srovnávání a studie neznámých algoritmů. Lze s ním analyzovat prvky v neznámém algoritmu a ty potom klasifikovat. Podle tutoriálu k programu se jeví tento program velice jednoduše, což se však může odrazit v úspěšnosti odhalení hledaných srdečních poruch. Tento systém se tedy hodí na hodnocení méně závažných signálů. Zmíněný systém je též úzce spojen s metodou rozhodovacích stromů. Tato metoda je založena na logické operaci disjunkce získané z konjunkce získaných hodnot(EKG) a vzoru. Výsledkem je „ano“- hodnoty odpovídají vzoru, nebo „ne“ –hodnoty jsou odlišné od vloženého vzoru. Spojení systému s touto metodou dosahuje 87% úspěšnosti.

Dalším a mnohem známějším systémem vhodným mimo jiné i pro klasifikace signálů je MatLab. Toto programovací prostředí budu i já využívat pro klasifikaci. Jedná se o matematický program vhodný ke všemožným výpočtům a tvorbě prostorových grafů a jeho novější podoby již dovolují i tvorbu graficky přehledných modulů využitelných třeba k mé klasifikaci signálů EKG. V MatLabu již lze použít více metod klasifikace signálů. Neznámý signál si nejprve můžeme vyfiltrovat pomocí FIR filtrů rychlé Furierovy transformace. Pomocí této filtrace získáme čistě užitečný signál připravený na klasifikaci. Ten pak můžeme klasifikovat opět metodou rozhodovacích stromů, ale již s vyšší přesností, nebo je zde možnost přímo provést spojitou vlnkovou transformaci CWT, která nám může dát výsledek jako míru podobnosti mezi databázovým vzorovým signálem a analyzovaným signálem. Další možností klasifikace je využití metody Dynamic Time Warping (DTW). Překladem Dynamic Time Warping je borcení časové osy. Metoda je založena na skocích po časové ose za účelem získání shodných tvarů signálů a za výsledek se vezme ta perioda, která se nejčastěji opakuje v signálu. U EKG signálu vybírá tato metoda periody od kmitu R po další kmit R. Tato vyhodnocená nejčetnější perioda signálu je rozkouskována na vlny P a T a na celek s názvem QRS(vysvětlení níže). Ty jsou potom vyhodnoceny srovnáním s databázovými údaji jednotlivých segmentů zvlášť. Na obrázku 8 je naznačeno řešení EKG signálu přes DTW metodu. Jedná se o trojrozměrnou distanční matici utvořenou z vybraného signálu a sloužící ke klasifikaci.

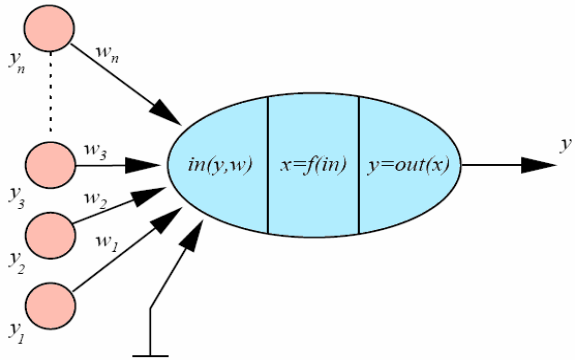


Obr.8 DTW metoda klasifikace

Mezi další metody lze zařadit i metodu nazvanou Hidden Markov Models (HMM). Její definice je dosti složitá, ale jednoduše lze říct, že se jedná o metodu konečných náhodných procesů.
Celé HMM jsou řetěz odpovídající konečným náhodným procesům s deterministickou výstupní funkcí b(q)=q. Upravený signál vstupuje přes tzv. tréninkový blok do konečného počtu procesových bloků, ve kterých se provede výpočet pravděpodobnosti. Tato metoda lze použít pro klasifikaci EKG, ale je daleko vhodnější ke hledání v DNA kódech.

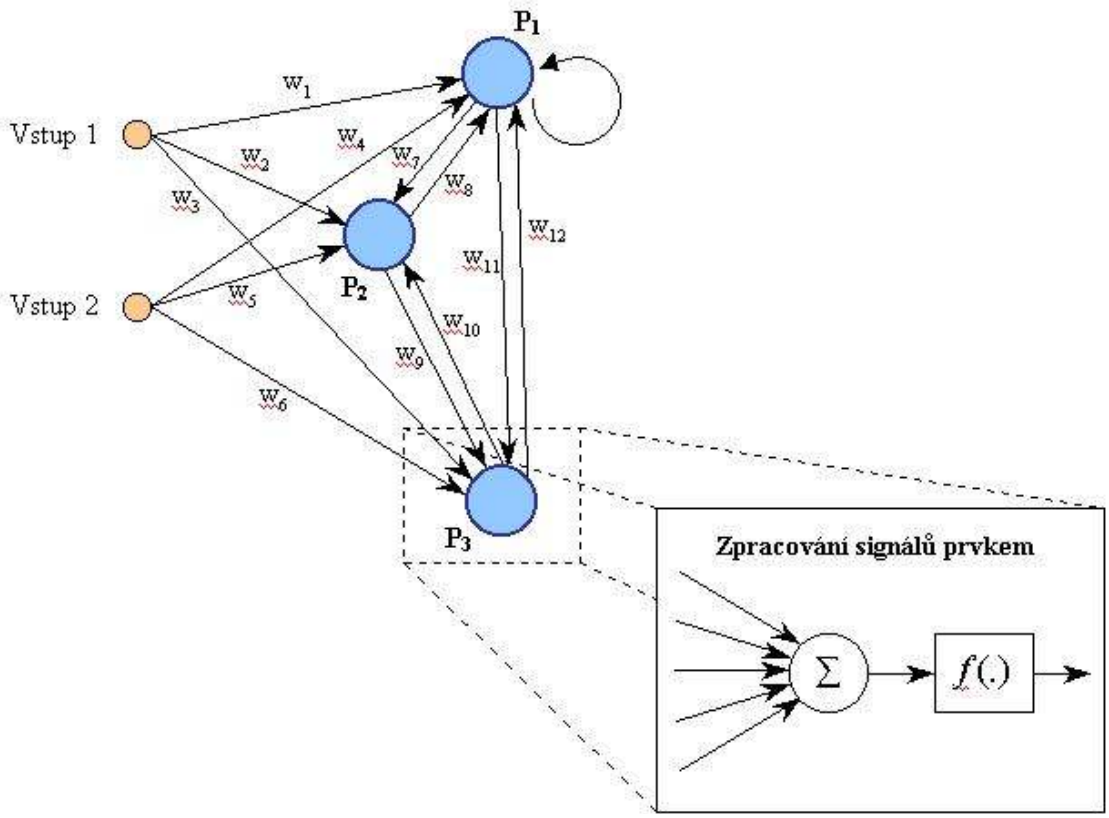
Poslední zde uvedenou metodou klasifikace signálu je metoda klasifikace EKG signálu pomocí umělých neuronových sítí. Podrobnější popis metody tohoto typu bude také obsažen v samostatné kapitole. Dalšími metodami se již tato práce nezabývá.

2 Neuronové sítě a klasifikace EKG :



Obr.9 Struktura jednotky sítě (umělého neuronu)

Umělé neuronové sítě (Artificial Neural Network) vznikly modelací biologických sítí buněk mozku. Svého využití našly v elektronice, automatizaci, chemii, informatice a dokonce i ekonomice. Sítě jsou složené z mnoha procesorových jednotek mezi sebou paralelně propojených informačními kanály. Tyto sítě mají zavedený nový pojem „učení/trénink“. Tento pojem označuje schopnost sítě přijatý signál zaznamenat do synaptických vah a další přijatý signál s takovýmto signálem třeba ve fázi testování porovnat. Sítě mohou být různých architektur od jednoduchých dopředných sítí po různě složité rekurentní sítě. To nám definuje topologie sítě.
Hodnoty signálů přenášených mezi jednotlivými prvky se mění v závislosti na vahách w_{ij}. Prvek(umělý neuron) vytváří sumu všech příchozích hodnot vážených spojení a vytváří výsledek, který je nelineární (statickou) funkcí jeho součtu. Výstup prvku může být výstupem systému nebo může být předán jinému neuronu. Obrázek 10 je schématické znázornění umělé neuronové sítě se třemi neurony P, kde každý neuron, hlouběji popsaný na obrázku 9, vytváří výstupní hodnotu díky sumě vážených vstupních hodnot.



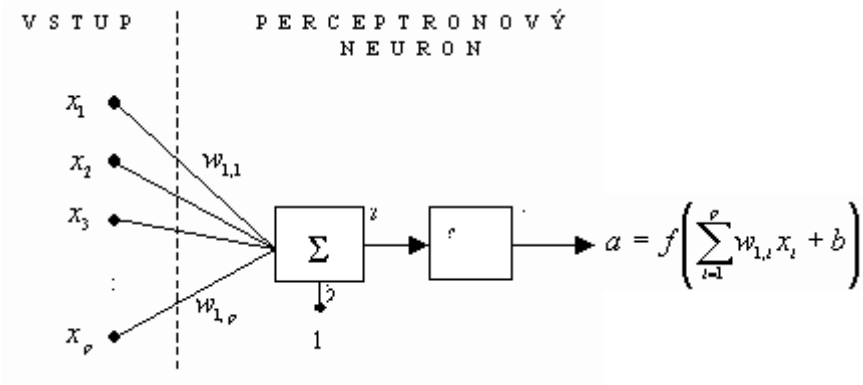
Obr.10 Schéma umělé neuronové sítě

V dnešní době existuje několik druhů sítí. Mezi nejpoužívanější patří sítě :

- Perceptronové
- Radiální
- Rekurentní

Ke klasifikaci EKG signálu nám bude stačit hlouběji prozkoumat funkci jednoduché vícevrstvé sítě s dopředným řízením. Při digitalizaci signálu budeme požadovat dostatečný počet vzorků a musíme počítat se stejným počtem vstupních neuronů a z tohoto počtu se následně odvíjí i počet klasifikačních tříd.

Použita je Perceptronová dopředná síť. Základní stavební jednotkou perceptronové sítě je perceptron zobrazený na obrázku 8, kde x jsou vstupy, p je počet vstupů, w jsou váhy spoje, b je prahová hodnota, f je aktivační funkce . Výstup perceptronu je označen a .



Obr.11 Schéma Perceptronu

Každý vstup je v perceptronu ohodnocen příslušnou vahou. Ta souvisí s daným spojem a suma popsaných vstupů je předána jako vstupní informace aktivační funkci, jejíž výstupem je výstup perceptronu. Výstupní hodnoty jsou v závislosti na vstupních hodnotách omezeny v rozsahu 0 nebo 1. V případě, že je prahová hodnota 0 to znamená, že pro všechny záporné vstupní hodnoty

bude na výstupu neuronu 0 a pro všechny nezáporné hodnoty bude na výstupu neuronu hodnota 1. Prahová hodnota způsobuje pouze posun vstupní hodnoty, kdy dochází ke změně výstupní hodnoty z 0 na 1. Režim učení vícevrstvých perceptromových sítí je spuštěn pomocí algoritmu zpětného šíření. Nejčastěji využívaným algoritmem pro trénování je Levenberg-Marquardtův algoritmus.
Použitím této metody můžeme získat až 99% úspěšnost při klasifikaci. K takovéto úspěšnosti však potřebujeme síť s velkou kapacitou, která bude mít naučených alespoň 10 000 vzorů .

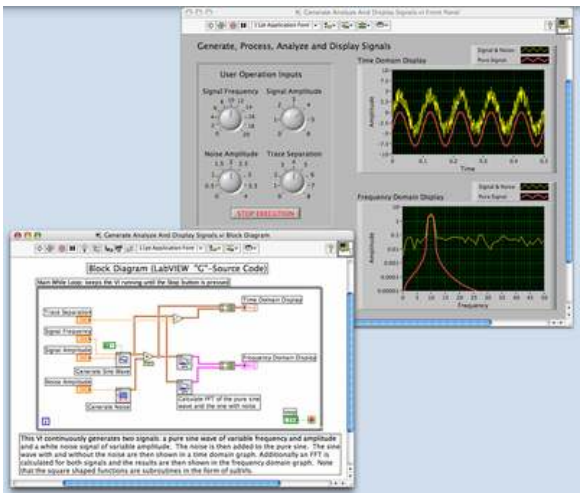
3 Výběr prostředí

V dnešní době existuje veliké množství méně známých vývojových prostředí. Mezi nejznámější patří prostředí jazyka C, kterým lze naprogramovat i to, s čím autoři jazyka vůbec nepočítali. Programovacím jazykem C lze bez větších problémů vytvořit grafický program fungující přesně dle našich představ. Není třeba vyrábět kompromis mezi ideálem a realitou. Ze zcela opačné strany šli k uživateli programátoři ze společnosti National Instruments. Jejich aplikace LabView umožní i základnímu uživateli vytvořit aplikaci z předem připravených bloků matematických i grafických. Dále se v dnešní době setkáme na každém kroku s programem společnosti The MathWorks. Jejich aplikace s názvem MatLab slouží k matematickým experimentům a výpočtům jevů všeho druhu. V současnosti je však z tohoto programu vytvořeno ideální prostředí pro tvorbu vlastních programů.

3.1 Prostředí jazyka C :

Jazyk C vznikl v 80.letech pod rukama Bjarne Stroustrupa v laboratořích AT&T Bell Laboratories.Byl vyvinut pro tvorbu operačního systému UNIX .Následně byl upraven a zpřístupněn odborné veřejnosti, která ho do dnes využívá. V dnešní době nabízí práci s čísly i objekty. V roce 1983-1985 vznikla verze C++, která měla za cíl rozšířit možnosti a zjednodušit samotné programování. V tomto jazyce lze dosáhnout až ideálních detailů při tvorbě grafických rozhraní. Jeho neustálý rozvoj je skryt v rostoucím počtu knihoven pro programování. Ty jsou ukryty v hlavičkových souborech tvaru *.h a stačí se na ně v průběhu programování jen odkazovat. I tak stále zůstává programování v tomto jazyce v rukách profesionálů a je bráno za dosti složité a zdlouhavé. Výsledky jsou však vždy dokonalé.

3.2 Prostředí LabView :



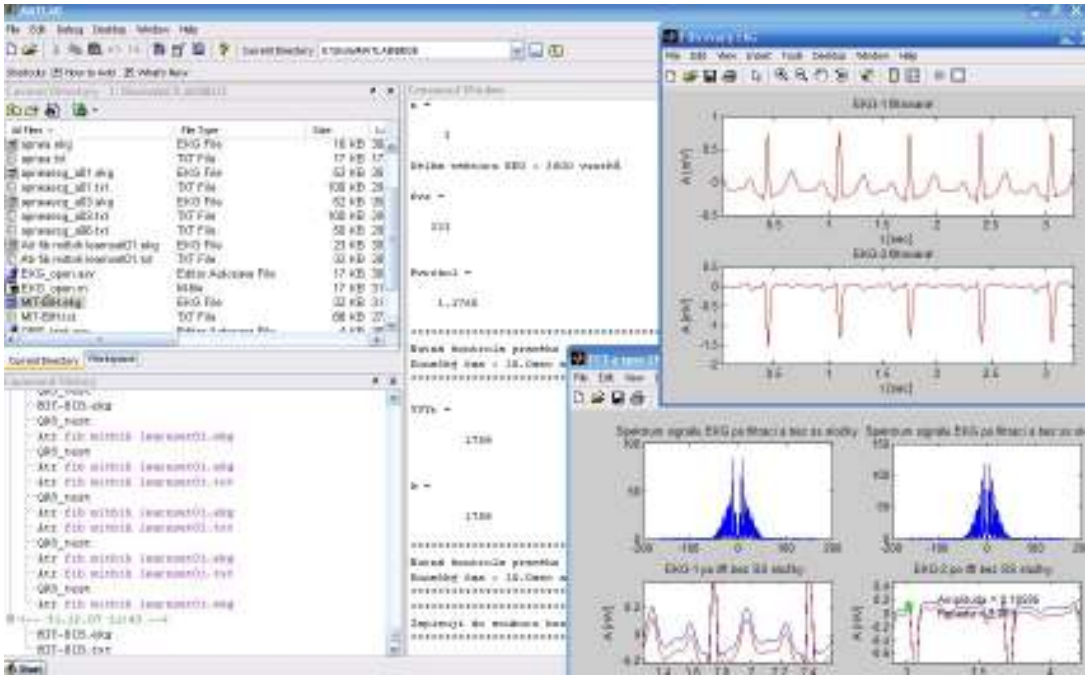
Obr.12 Ukázka prostředí LabView

LabView je celým názvem **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench. Jedná se o experimentální virtuální laboratoř. Je postaven na základě vizuálního programovacího jazyka. Jednotlivé matematické i datové operace provádíme v přehledných blocích, které mezi sebou spojíme. Každý důležitý blok nám automaticky ve vedlejším okně vytvoří svoji grafickou podobu. Aplikaci lze tvořit i opačným směrem, kdy na pole programu rozmístíme grafické tlačítka, potenciometry a grafy. Následně po přepnutí do pole bloků lze jednoduše bloky propojit dle svých představ. Během několika minut jsme schopni například vyfiltrovat a

zpracovat EKG signál a následně jej uložit do souboru. Navíc je aplikace od prvopočátku uživatelsky přívětivá, přehledná a jednoduchá na ovládání konečným uživatelem. Její nevýhodou je, že pokud postrádá nějakou funkci, nebo funkční blok ,musíme si danou funkci velice složitě vytvořit a nemusí se nám to ani povést. LabView je ze sledované skupiny nejmladší prostředí a je tedy možné, že v průběhu vlastního vývoje časem předčí jiné matematické, či experimentální prostředí.

3.3 Prostředí MATLAB :

Společně s jazykem C patří mezi nejstarší programovací prostředí a vznikl v 70.letech. Jeho vývoj však pod křídly společnosti The Mathworks začal až v roce 1984. Původně sloužil k čistě matematickým výpočtům a experimentům, ale postupem času stejným způsobem, jako prostředí C na sebe nabaloval knihovny zvané Toolboxy. Pomocí níž lze nahradit složité rovnice a výpočty jednoduchým příkazem a doplněním jeho parametrů. Například již není nutné vkládat přesné rovnice rychlé Fourierovi transformace. Stačí pouze napsat zkratku fft s parametry a transformace se nám automaticky vypočítá. Matlab též spojuje výhody obou dříve jmenovaných prostředí. Poradí si s každou možnou operací a lze v něm tvořit také grafické bloky pro jednoduchou práci a tvorbu aplikací. Dá se považovat za profesionální a zároveň přijatelně složité programovací prostředí. Právě z těchto důvodů byl Matlab vybrán pro tvorbu aplikace na úpravu a klasifikaci elektrokardiogramu. Obsahuje přímo Toolbox pro tvorbu umělých neuronových sítí a nepřeberné množství přípravků pro filtraci signálů.



Obr.13 Ukázka prostředí Matlab

4 Ideální a reálná podoba programu

V lékařské praxi patří mezi hlavní kritéria rychlost a přesnost určení diagnózy. K tomu může pomoci lékaři vytvořená aplikace v duchu této studijní práce. Samotný program musí být pro lékaře naprosto přehledný a přitom musí obsahovat veškeré potřebné údaje. Celý děj programu by se měl odehrávat na jediné obrazovce, aby lékař nemusel složitě dohledávat stránku, na které je zobrazen hledaný údaj, či zobrazen zkoumaný elektrokardiogram. Navíc je nutné, aby jednotlivé ovládací prvky byly dostatečně veliké. Na splnění požadavků by jsme potřebovali velikou obrazovku a to by zase vedlo k nepřehlednosti a lékař by musel otáčením celé hlavy dohledávat požadované prvky programu. Je tedy patrné, že není jednoduché poskytnout lékaři ideální aplikaci. Pokud tedy chceme poskytnout uživatelsky přívětivý program, musíme do určité míry zredukovat zobrazované údaje a možnosti, nebo rozložit program do dvou přepínatelných obrazovek. Aby se lékař naučil s programem pracovat, je pro něj vhodnější postupné rozšiřování programu tak, jak tomu je v běžné praxi od operačních systémů po řadové programy na přehrávání videa. První operační systém uměl pouze pár příkazů stejně tak, jako první přehrávač videa dokázal video pouze otevřít, pustit a vypnout. Pro tvorbu uživatelsky přívětivého programu se tedy budeme držet téhož postupu. a až v další verzi lze přistoupit na dvou-obrazovkové provedení. V první verzi se pokusíme zobrazit celý průběh vloženého a zpracovaného signálu, lupu pro zobrazení libovolného detailu signálu, výpis získaných dat včetně výpisu zjištěných abnormalit v signálu, seznamové menu pro volbu zobrazení patologických impulsů v signálu, zatržítka pro zobrazení signálů s různou filtrací, dodatkové menu pro volbu drobností a hlavně tlačítka pro ovládání programu. Pro první verzi volíme sestavu tří tlačítek s funkcí pro otevření signálu, uložení získaných dat do souboru a hlavně tlačítko pro spuštění úprav včetně klasifikace. Tlačítko určené pro otvírání souborů lze časem udělat automatické, ale pro experimentální fázi projektu je důležité. Nyní je potřeba se zamyslet nad ideálním rozvržením výše uvedených bloků a jejich velikost. Za nejvýznamnější bloky lze považovat zobrazovače průběhů (grafy), pole získaných parametrů i patologických dat a hlavně tlačítko, které vše spustí. V dnešní době se totiž běžně setkáváme s programy, kde lze nastavit velké množství vstupních dat například kompresních metod a poměrů, ale nejtěžší je najít tlačítko, kterým se celá akce odstartuje. Podobný případ je zobrazen na obrázku 4 nejmenovaného profesionálního kompresního programu. Je zde sice patrné tlačítko pro spuštění komprese videa, avšak marně může uživatel hledat tlačítko pro vložení vstupního souboru s videem ke komprimaci. Toto tlačítko tam zcela chybí a originální video pro komprimaci lze načíst pouze klepnutím pravého tlačítka myši do okna nad ovládacími prvky dole. Tento nešvar způsobil, že jeden z nejlepších kompresních programů není uživateli pochopen a je na softwarovém serveru hodnocen negativně, až zcela záporně.

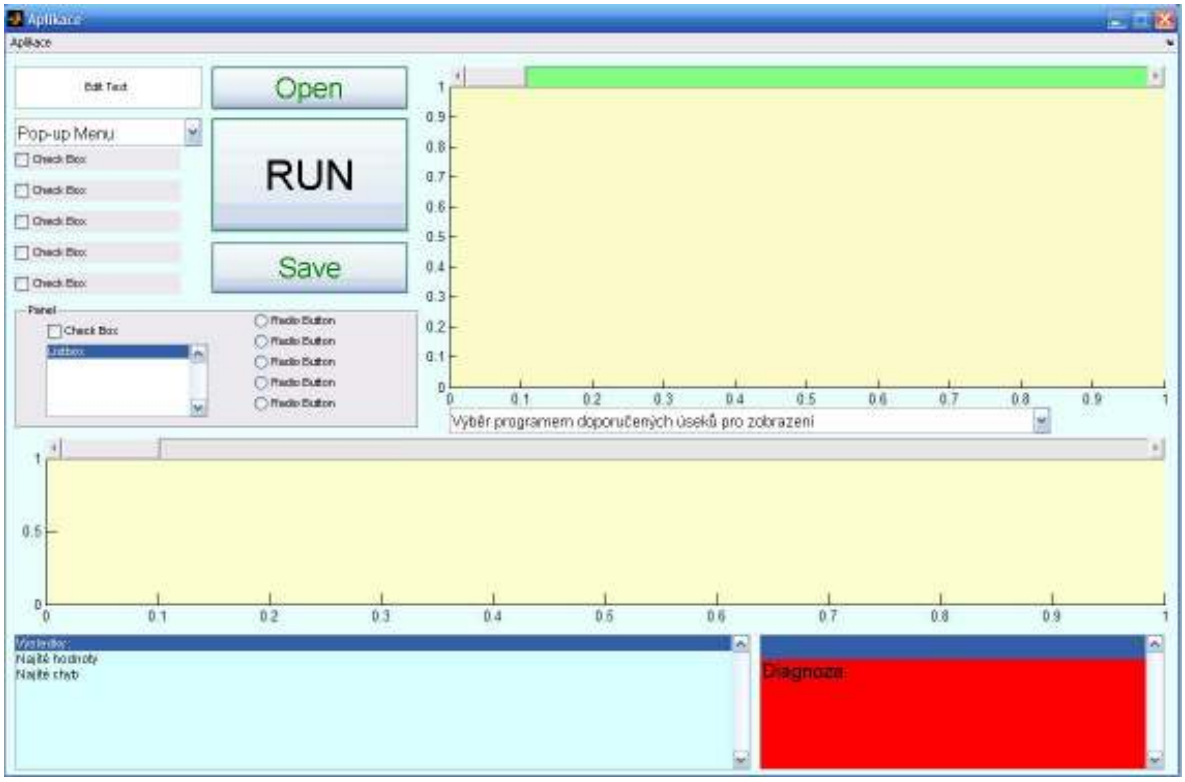
Obr. 14 Špatně navržený program

Dále má člověk vžitý postup, jehož se během užívání programu drží. Je tedy třeba uspořádat ovládací prvky na jedno místo v posloupnosti vyhovující většině uživatelů. Teorii ideálu již známe a nyní si zobrazíme několik možností rozvržení. Ty byly zhotoveny přímo v prostředí Matlab pomocí utility „GUIDE“, což je Graphical User Interface development environment, nebo-li uživatelské grafické vývojové prostředí.

4.1 Finální návrh vzhledu aplikace :

Hlavní ovládací prvky jsou umístěny nahoru, avšak jsou již uspořádány tak, aby se veškeré nastavení odehrávalo v jednom místě. To má v uživateli budit dojem, že mu program přímo nabízí postup pro úspěšnou klasifikaci. Největší část obrazovky zde zabírá okno se zobrazením delšího intervalu bez detailu. Okno Lupy je umístěno vpravo nahoře. Jeho schopností je zobrazit v něm dostatečný detail impulsu. Velice vhodně je volena pozice pro výběr a zobrazení patologických jevů. Za tímto účelem byla i umístěna konečná klasifikace, kterou by dle průzkumu v pravém dolním rohu hledala většina dotázaných uživatelů.

Konečná podoba návrhu vytvořená podle několika předloh je zobrazena na obrázku 15 a na jejím základě by měl vzniknout výsledný program.



Obr. 15 Finální návrh

5 Zpracování signálu EKG

Analogový signál je nejprve převeden na digitální a společně s údajem času je uložen v matici hodnot. Postupu digitalizace se již nebudu věnovat, protože z experimentálních databází již získáváme matici navzorkovaných dat signálu. Ty se však nedají přímo použít a je nutné jejich tvar upravit do podoby použitelné pro další úpravy. Za tímto účelem vznikla úvodní aplikace, která si nejprve načte jakýkoliv soubor do své paměti a následně v něm hledá uspořádání čísel do řádků, či sloupců. Pomocí příkazu length nejprve zjistíme počet sloupců a následně pomocí cyklu převedeme jednotlivé sloupce do vektoru času a napětí signálu. Při tom však očekáváme, že údaje v prvním sloupci jsou vždy údaje o času. K zajištění toho, že tomu tak opravdu je, by stačila jednoduchá podmínka pro zjištění, zda hodnoty ve sloupci neustále narůstají. Z údaje času si nyní vypočteme frekvenci vzorkování, jež se pohybuje od 333Hz po 1000Hz a určíme si počet řádků ve sloupci. Tyto získané údaje nám slouží k rozhodnutí pro kompresi dat a základní filtraci vysokých frekvencí způsobem vynechávání každého druhého bodu vektoru. Díky této úpravě signálu se o polovinu zkrátí veškeré výpočty, což je velice důležité pro celkovou dynamiku programu. Zároveň tím eliminujeme vysoké frekvence, avšak tento jednoduchý a běžně nepoužívaný způsob úpravy signálu vede k zeslabení R vlny, která většinou bývá celá obsažena v pěti vzorcích. Pokud tedy touto metodou odstraníme údaj obsahující maximální hodnotu, klesne tím amplituda R vlny. Po řadě pokusů se však tento jev příliš nevyskytl a všechny R vlny byly programem správně vyhledány. Tím tedy klesne vzorkovací frekvence na polovinu. Pokud by následně byla příliš nízká vzorkovací frekvence, tak se tato operace úpravy automaticky nepoužije. Dále je nutné provést filtraci pro odstranění pohybových artefaktů a parazitních frekvencí, mezi něž patří i frekvence sítě 50Hz. Pásmo čistého signálu EKG se pohybuje od 3 do 45Hz. Nižší frekvence způsobují pohybové artefakty a vyšší frekvence tvoří různé zákmity a „chvění“ signálu. Pro filtraci signálu lze všeobecně použít veliké množství metod od různých kumulací po Fourierovu transformaci a diskrétní vlnkovou transformaci. Ne všechny však můžeme použít pro filtraci našeho signálu, protože by jsme si mohli například kumulací vyrušit některé patologické jevy, které máme v dalším bodě klasifikovat. Mezi vhodné metody patří diskrétní vlnková transformace, filtry s konečnou impulsovou odezvou a rychlá Fourierova transformace. Za nejvhodnější je považována právě diskrétní vlnková transformace. V mém případě je k úpravě volena jednodušší metoda filtrace signálu. Nejprve jsou použity filtry FIR typu pracující dle vztahu :

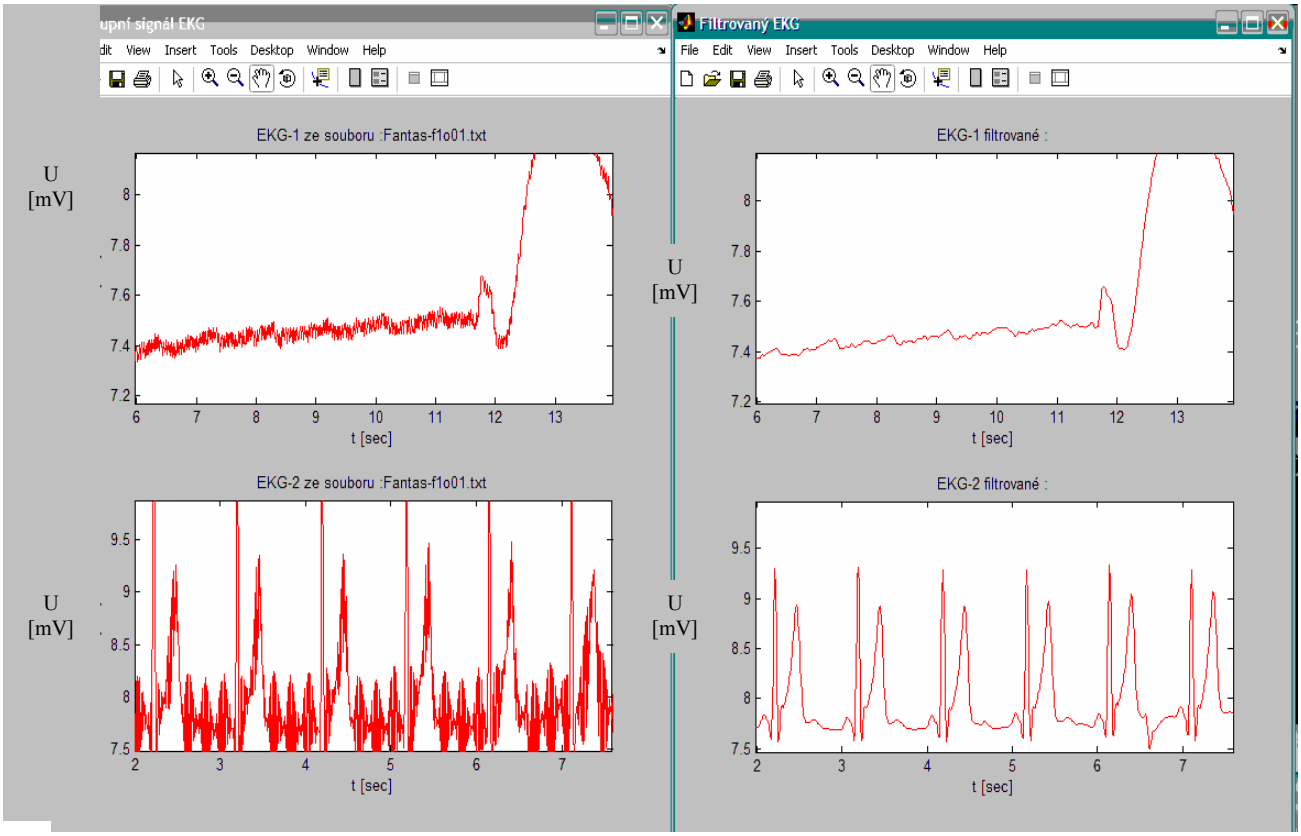
$$y[n] = b_0 x[n] + b_1 x[n-1] + ... + b_N x[n-N],$$

kde y a x jsou výstupní a vstupní hodnoty n-tého vzorku a N je počet koeficientů,nebo-li řád filtru. K dosažení dostatečné filtrace signálu touto metodou, je třeba volit vysoký řád filtru. To se však negativně projeví na zpoždění signálu. Zpoždění lze eliminovat posunem celého filtrovaného signálu po časové ose. Tato operace nemůže klasifikaci ohrozit, avšak reálně toho dosáhnout nelze a eliminace zpoždění se provádí naopak umělým zpožděním originálního signálu. V prostředí Matlab lze filtr navrhnout jednoduše pomocí příkazu :

$$B = \text{fir1 (N, [Vf}_{DOLNI}\ \text{Vf}_{HORNI}\text{ , ‘typ‘]) ;}$$

kde N je řád filtru, [Vf_{DOLNI} Vf_{HORNI}] je vektor dolní a horní mezní frekvence a nakonec typ je druh filtru. V našem případě pásmová zádrž. Volbu řádu filtrů jsem provedl experimentální metodou a vektor frekvencí jsem určil výpočtem $Vf=f_m / f_{vz}$.

Odstraněny byly tedy frekvence do 3Hz, v okolí 50Hz a frekvence vyšší. Pro dosažení menšího zpoždění a přitom dobré filtrace byl signál prohnán jedním filtrem vícekrát za sebou. Přesto však po filtraci zůstává v signálu téměř nezměněná ss složka a bylo třeba zcela odstranit vyšší frekvence. Filtraci lze posoudit na obrázku 16 s originálním a filtrovaným signálem. Je zde patrný pokles amplitudy R vlny. V této míře to však na funkci celého programu nemá skoro žádný vliv.



Obr.16 Vlevo originální signál a vpravo filtrovaný signál EKG pomocí FIR filtrů

K odstranění stejnosměrné složky a vyšších kmitočtů byla použita rychlá Fourierova transformace (FFT) . Ta je založena na transformaci vektoru signálu na vektor frekvenčního spektra

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

kde x je vektor signálu , X je vektor spektra frekvence a ω_N je

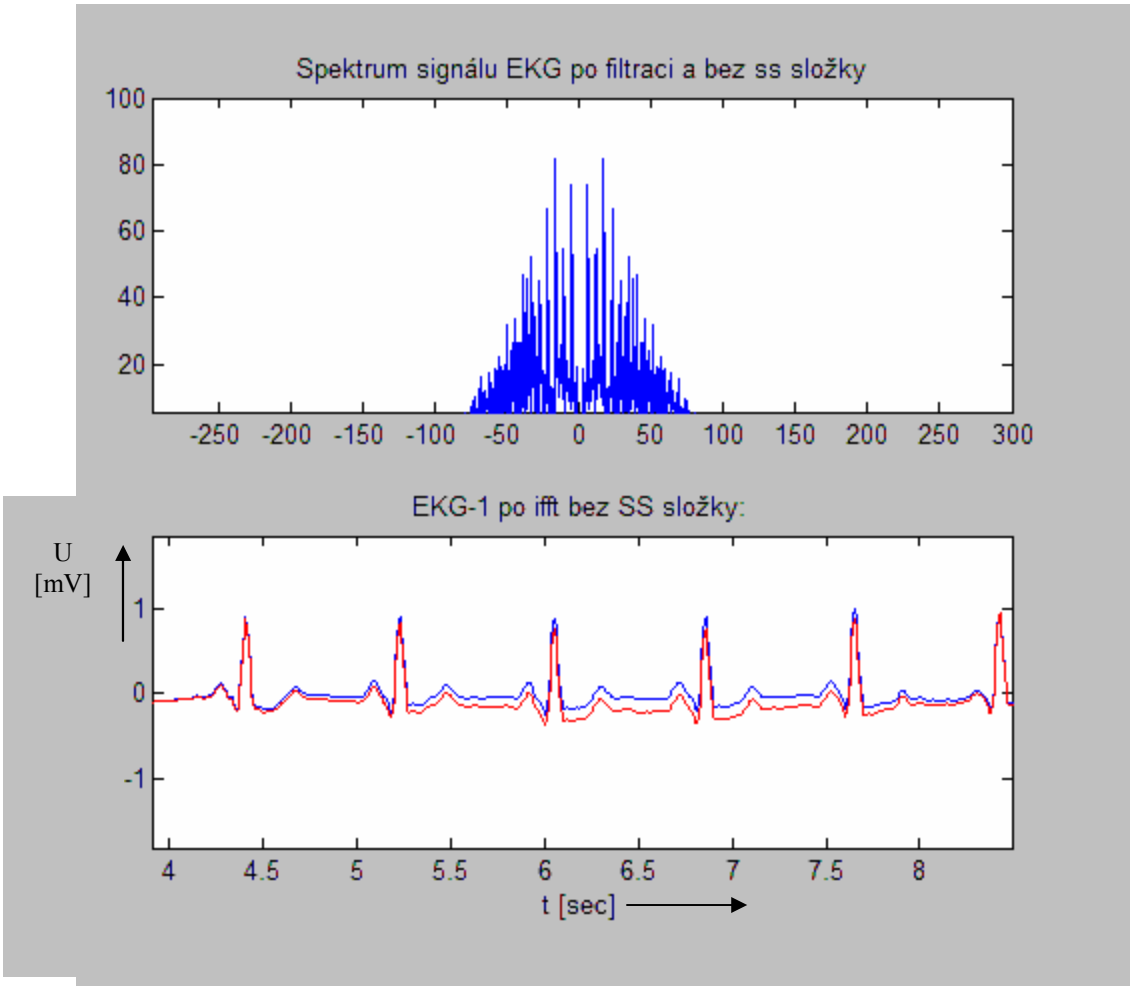
$$\omega_N = e^{(-2\pi i)/N}$$

kde pomocí N určíme vzdálenost zkoumaných sousedních frekvencí a frekvence umístěné uvnitř vzdálenosti budou ignorovány. Například pro N = 2 získáme z celého vektoru signálu jen dvě frekvenční spektra, ze kterých při zpětné Fouriérově transformaci nemůžeme získat jiný, než téměř harmonický signál. Za N tedy dosazujeme řádově stovky. V prostředí Matlab se tato transformace provádí následujícím příkazem : Y = fft (X , n) ; kde Y je výsledný vektor „četnosti frekvence“ (V / Hz), fft je příkaz pro užití rychlé Fourierovi transformace, X je vstupní EKG signál a n je výsledný počet frekvencí, na kterých je zkoumána četnost. Pokud N není zadáno, tak se zkoumá četnost všech frekvencí. V tomto spektru můžeme nyní provést filtraci způsobem, kdy vynulujeme nežádoucí frekvence. Pro nás je to tedy nežádoucí první spektrální čára, která obsahuje nulovou frekvenci (stejnosměrnou složku). Zde se nám též mnohem jednodušeji a účinněji podaří odstranit pohybové artefakty v signálu. Pokud frekvenční spektrum obsahuje i velkou část vyšších frekvencí, tak je pomocí podmínky též ze spektra odstraníme.

Vektor spektra následně převedeme zpět na vektor signálu pomocí zpětné Fourierovi transformace označované jako iFFT (inverzní rychlá Fourierova transformace). Ta má následující tvar :

$$x(j) = (1/N) \sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

V prostředí Matlab se provádí příkazem : y = ifft (X , n). Na následujícím obrázku s číslem 17 je zpracovávaný další signál. V horní části je zobrazeno spektrum s odstraněnými nízkými frekvencemi, které nyní tvoří díru ve středu spektrálního grafu, tedy v nule a jejím okolí je vidět světlý zub. Ve spodní části obrázku je zobrazen výsledek zpětné Fourierovi transformace modře a s ním je zobrazen červeně jen FIR filtry filtrovaný signál. Je zde vidět, že pásmová zádrž FIR filtru není tak účinná, jako nulovaní frekvenčních spekter pomocí Fourierovi transformace.



Obr.17 Nahoře frekvenční spektrum signálu s odstraněnými spektrálními čarami, dole modře výsledek FFT a červeně signál získaný pouze filtrací FIR.

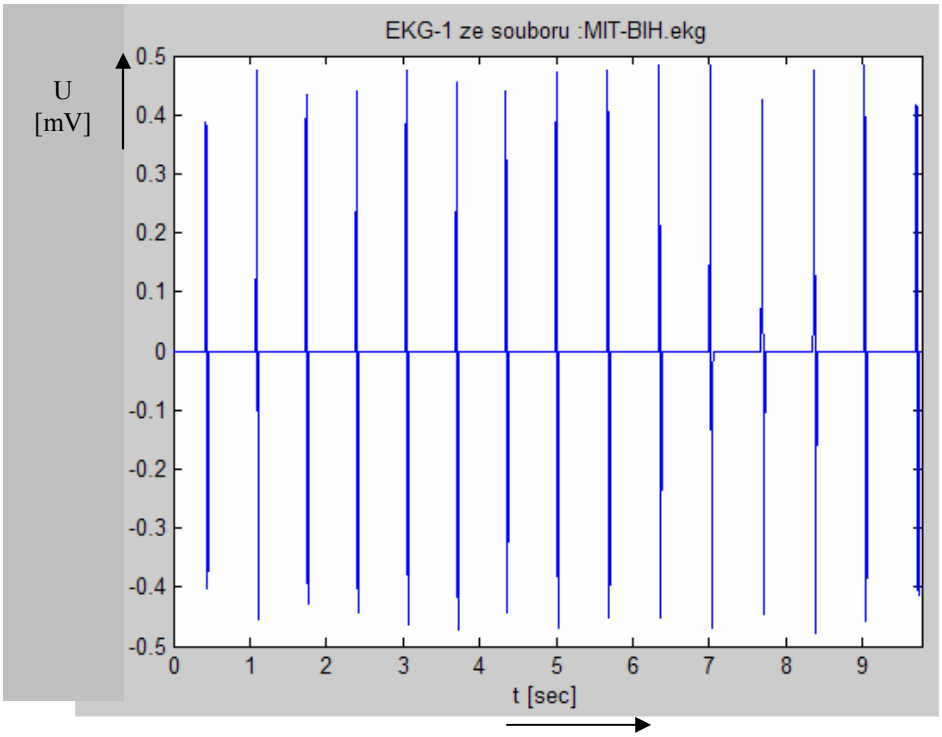
Nyní je signál dostatečně zpracovaný pro klasifikaci. Celé zpracování signálu je vytvořeno a uloženo v jednom m-souboru programu Matlab s názvem „EKG_open“ a je v příloze. Tato základní procedura programu je otestována na velkém množství signálů. Ve většině případů jsou výsledky výborné pro klasifikaci, ale u úprav některých speciálních signálů docházelo k jejich deformaci. K tomu dochází u signálů se zcela odlišnou vzorkovací frekvencí, nebo signálů s velice úzkou vlnou R a naopak relativně vysokou a širokou vlnou T. K odstranění vad lze nejpravděpodobněji použít jiné metody filtrace, například vlnkové transformace popsané Doc. Ing. Jiří Kozumplíkem , CSc. v knize *Vlnkové transformace a jejich využití pro filtraci signálu EKG* .

Výsledek, tedy zpracovaný signál a jeho časování, je uložen do souboru původního názvu s koncovkou EKG(*.ekg) . Při každém spuštění výše popsané procedury se soubor s koncovkou ekg automaticky zaktualizuje.

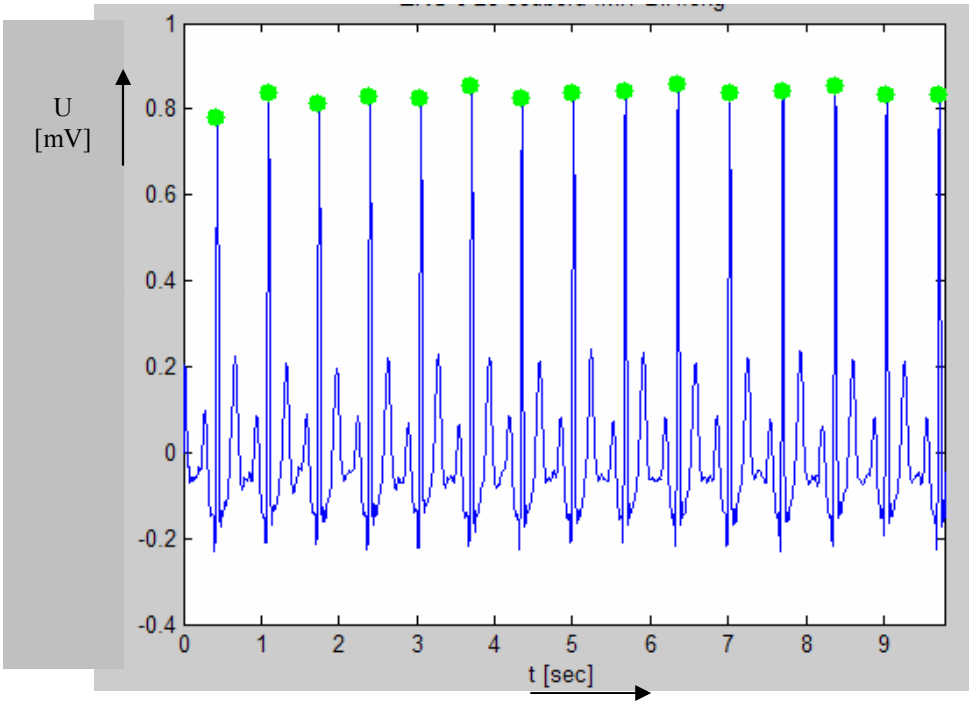
5.1 Příprava signálu pro klasifikaci

Pro přípravu signálu na klasifikaci byl vytvořen další m-soubor s novou procedurou. Jeho název je „QRS_test.m“ a jeho výpis je součástí přílohy. Zpracovaný signál ze souboru s koncovkou EKG si otevře a načte do paměti pod novým označením. Po prvotní přípravě program přechází k detekci QRS komplexu, což je spojení tří vln s názvy Q, R a S podrobně popsaných v úvodní části práce. Nejjednodušší a překvapivě dokonale pracující je metoda založená na hledání bodů ležících nad zvolenou, popřípadě vypočtenou úrovní. Metodu lze pojmenovat jako hledání QRS komplexu pomocí amplitudy R vlny. Úroveň, podle které o existenci R vlny rozhodujeme, může být buď pevně experimentálně zvolena, nebo může být vypočtena součinem maximální hodnoty zkoumaného signálu a experimentálně zvolenou konstantou v našem případě 0,3. Tato metoda je funkční do doby, než se v signálu vyskytne artefakt převyšující největší R vlnu, nebo se amplitudou této vlně přibližuje. Na eliminaci artefaktů převyšujících největší R vlnu lze použít další ze způsobů určení úrovně. Výpočtem si zjistíme průměrnou hodnotu celého elektrokardiogramu a k této hodnotě pohybující se v blízkosti nuly přičteme experimentálně zvolenou konstantu. Zaznamenáme nyní tedy všechny QRS komplexy, avšak navíc i další artefakty označíme za QRS komplex, což je pro lékaře zavádějící. Některé artefakty označené za R vlnu lze eliminovat pomocí funkce derivace. Základní vyjádření derivace je jako poměr, v jakém růst nějaké závislé proměnné y odpovídá změně jiné proměnné x. Jde tedy o míru změny funkce v daném bodě $\frac{\Delta y}{\Delta x}$.

Za nejefektivnější je po řadě experimentů považováno spojení metod 1. derivace a amplitudové hledání R vlny. Nejprve ze signálu odstraníme záporné hodnoty vynásobením (-1) a následně pomocí vypočtené úrovně hledané amplitudy odstraníme nižší hodnoty amplitud a provedeme derivaci takto upraveného signálu. Výsledek derivace elektrokardiogramu je na obrázku 18. Tento vektor hodnot však může obsahovat i jiné vlny, než jen vlnu R. Pro jejich eliminaci je použita výše popsaná metoda hledání bodů ležících nad zvolenou úrovní. A protože jedna R vlna může sestávat z více bodů a mohlo by se stát, že byla jedna R vlna zaznamenána jako pět R vln, je třeba po zjištění vyhovujícího bodu přičíst k cyklu hodnotu konstanty a tím přeskočit další body dané R vlny. Ta byla pokusy stanovena na hodnotu 80. Při této hodnotě by neměly být zanedbány dvě R vlny blízko sebe. Amplitudy vyhovující všem podmínkám jsou následně zapsány do vektoru R-vln. Okamžité časy R vln jsou zaznamenány do dalšího vektoru. Tuto matici spojující oba vektory následně promítneme do elektrokardiogramu vstupního signálu. Výsledek je zobrazen na obrázku 19. Ze dvou sousedních časových údajů R vln získáme hodnotu RR intervalu odečtením bodu předcházejícího od aktuálního. Tím získáváme první údaj důležitý pro klasifikaci. Uvnitř RR intervalu pomocí hledání maxima a minima jsou nalezeny vlny T a S. Podmínkami a výpočty jsou získány vstupní hodnoty sítě. Ty jsou v prostředí Matlab označeny vždy trojicí stejných písmen od a do ch. Jedná se ve většině případech o průměrnou odchylku od normálu. Na základě těchto proměnných může probíhat samotná klasifikace.



Obr. 18 Elektrokardiogram po derivaci



Obr. 19 Elektrokardiogram se zobrazením bodů

5.2 Samotná klasifikace a umělé neuronové sítě

5.2.1 Pozorovatelné patologické jevy :

Můžeme již sledovat frekvenci tepu, amplitudu R vlny, jednotlivé RR intervaly a vynechaný, či zvýšený počet R vln ve zkoumaném časovém úseku. Z tepové frekvence lze určit, zda diagnostikovaný pacient vůbec žije, je v klidu, nebo zda má fyzickou zátěž. Pokud je frekvence tepu vyšší, než 90 tepů za minutu a sledovaný objekt nevykonává extrémně fyzicky náročnou činnost, klasifikujeme takový jev za tachykardii, klesne-li frekvence pod 55 tepů za minutu, hovoříme o bradykardii. U amplitudy RR signálu se pokoušíme vypozorovat kolísání její velikosti. Pro sinusové kolísání klasifikujeme výskyt elektrického alternansu, který často doprovází supraventrikulární tachykardii, popřípadě tachyfibrilaci síní. Jeho podoba je obecně naznačena na obrázku 20.

Obr. 20 Elektrický alternans

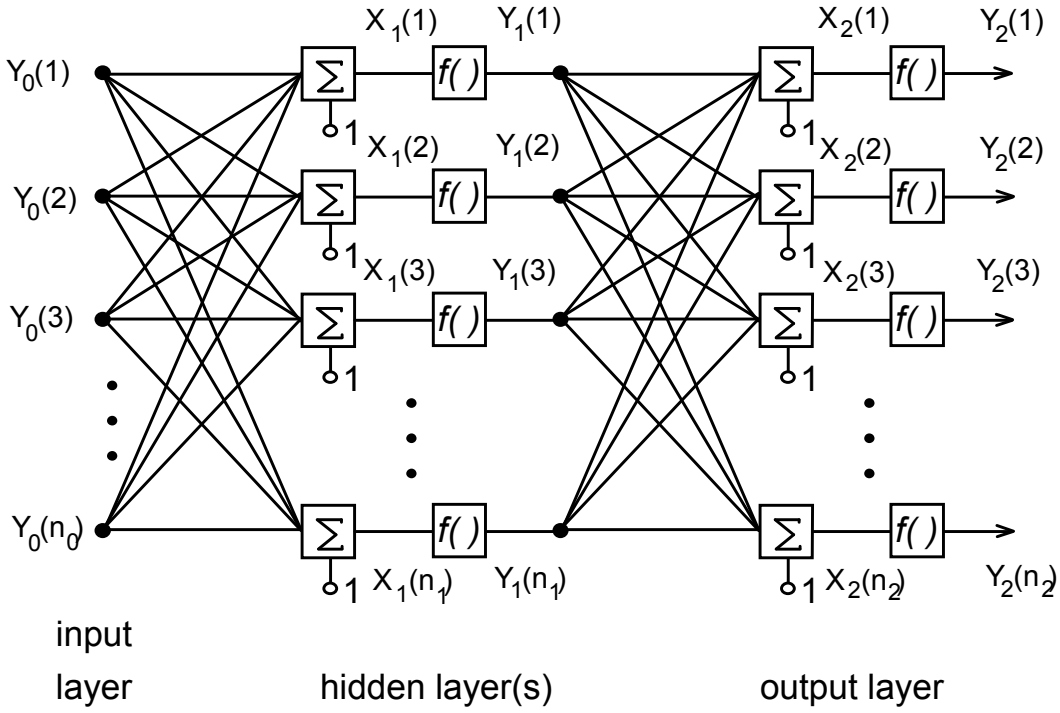
Z průměrného RR intervalu lze podmínkou zjistit, zda je RR interval patologicky změněn. Pomocí toho lze sledovat ischemii, arytmii a s ní spojenou fibrilaci síní. Z detekované T vlny lze zjistit z tvaru, posunutí a amplitudy hyperkalémii a hypokalémii. Hyperkalémie se projevuje vysokou a relativně úzkou T vlnou. Při hypokalémii dochází k posunu vlny T k dalšímu QRS komplexu a samotná vlna má nižší amplitudu a může již splynout s vlnou P. Dále lze najít vlnu T s opačnou polaritou. Podle záporného prokmitu lze prohlásit, že se může jednat postupně o myokarditis, perikarditis a infarkt myokardu. Kmit S z QRS komplexu je detekována přes minimum na pravé straně signálu do experimentálně zvolené vzdálenosti. K tomuto kmitu je dále ověřováno, zda signál nezůstává dlouze na úrovni vlny S a zda neleží uvnitř RR intervalu místo s nižší úrovní, než jakou má kmit S. Pokud by bylo nadetekováno setrvání na podobné úrovni po delší dobu, může se jednat o projev angíny pectoris v počáteční fázi. Pokud by se v RR intervalu blížícímu se dvojnásobku průměrné hodnoty nacházelo místo s nižší úrovní, je takový artefakt označen za extrasystolu. Spojením všech pozorovaných prvků lze detekovat deprese ST intervalu, zástavu dechu, zpomalený dech a částečně lze detekovat i tachykardie.

5.2.2 Nastavení a učení umělé neuronové sítě pro klasifikaci :

Při nastavení sítě je důležité správně zvolit funkci, podle které se hodnoty zpracovávají. Nejčastěji používané funkce jsou sigmoida, hyperbolický tangens, identita, saturace, signum a Gaussova funkce. Pro tuto úlohu je použita funkce sigmoida označená jako logsig. Analytické vyjádření této funkce je :

Sigmoida $$y = \frac{1}{1 + e^{-x}}$$

Volena je struktura dopředné sítě dvouvrstvé. K ní je potřeba vytvořit učící matici.



Obr. 21 Vrstvená perceptronová síť s jednou vnitřní vrstvou

Na obrázku je zobrazena použitá dopředná dvouvrstvá perceptronová síť. Sledovanými parametry jsou počet vrstev a počet neuronů v jednotlivých vrstvách. Počet vstupních neuronů je vhodné volit podle počtu vstupů vektoru. Při klasifikaci je použito 9 vstupních hodnot vektoru a je tedy voleno 9 neuronů v první vrstvě. Volba počtu neuronů ve vrstvě druhé lze volit od 1 do nekonečna. Neplatí zde však, že čím více neuronů bude použito, tím lepší výsledek dostaneme. Nejvhodnější je volba optimálního počtu. Při této klasifikaci je počet neuronů volen k počtu možností výsledků. Těch je předdefinováno 14. Počet výstupních neuronů je proto také volen na čtrnáct. Vstupní prvky jsou označeny jako vektor P, výstupní jako vektor T. Samotná modelace se provádí příkazem „newff“ :

```
net1 = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
```

kde net1 je vytvořená síť se vstupním vektorem P a počtem neuronů v první S1 vrstvě a druhé vrstvě S2 za použití sigmoidy u obou výstupních vrstev v síti. Síť je ještě nutné inicializovat například příkazem „init“. Po nastavení počátečních vah a prahů označených jako W a b program přechází k učení sítě pomocí učitele, kterého tvoří matice několika katalogově klasifikovaných signálů, podle kterých bude moct aplikace následně klasifikovat neznámé signály. Učení se provádí příkazem „train“ , kde na vstup je poslána matice učitele a na výstup je dodán předdefinovaný výsledek T :

```
[net2,tr] = train(net1,P,T);
```

Teoreticky by matice učitele měla obsahovat tisíce průběhů. V této práci je učební matice složena ze čtrnácti různých klasifikovaných signálů s různou diagnózou. Pro spolehlivější klasifikaci je vytvořena jedna matice s modifikovanými signály stejných typů. Obě matice jsou sítí při učení prohnány třikrát a navíc je při každém učení nastaven počet kroků učení (epochs) na alespoň 500 epochs a dále je nutné nastavit požadovanou kvadratickou chybu „goal“. Následné učení probíhá do doby ,než není kvadratická chyba nižší, než nastavená. Učení může být

ukončeno také překročením počtu kroků. Učení samotné zabere mnoho času a proto by bylo vhodné mít síť naučenou a pouze ji používat ke klasifikaci. Tento nedostatek lze odstranit použitím zatržítka, které v případě zatržení přeskočí opakované učení.

5.2.3 Zvolené třídy klasifikace :

Třídy jsou voleny podle kapitoly *1.3 Diagram pro zjednodušení klasifikace* . Ten byl vytvořen v rámci prvního projektu. Původně bylo předpokládáno, že bude síť klasifikovat pole fialové barvy. Ve finální podobě je však jeho klasifikace orientována spíše k síňovým arytmiím. Pomocí nedetekovaných odchylek od normálu může síť sledovat naučené třídy :

- Angína pectoris – deprese ST
- Extrasystoly
- Arytmie - porucha tvorby vzruchu - síňové a respirační arytmie
  - síňové fibrilace
- Plicní embolie
- Apnea-zadržený dech,pomalé dýchání
- Hypokalémie
- Hyperkalémie
- Perikarditis / Myokarditis
- Infarkt myokardu

Výstupem sítě je vždy jedna až tři věty z níže uvedeného seznamu vět :

```
veta{1,1}=('Extrasystola,příznak angíny Pectoris');%a
veta{2,1}=('Počáteční fáze/možnost výskytu angíny Pectoris. Jinak OK.');%b
veta{3,1}=('Respirační arytmie');%c
veta{4,1}=('Jemnovlnná fibrilace síní');%d
veta{5,1}=('Fibrilace síní(i hrubovlnná)');%e
veta{6,1}=('Zadržený dech');%f
veta{7,1}=('Podezření na plicní embolii,tachykardie');%g
veta{8,1}=('Pomalé,nebo zadržené dýchání,tachykardie');%h
veta{9,1}=('Fibrilace síní ');%i
veta{10,1}=('Hypokalémie');%j
veta{11,1}=('Hyperkalémie');%l
veta{12,1}=('Perikarditis / Infarkt Myokardu');%m
veta{13,1}=('Perikarditis / Myokarditis');%o
veta{14,1}=('Deprese ST intervalu => Angína Pectoris.');%p
veta{15,1}=('Slabá arytmie , OK ');%q
```

Součástí výsledku je i číselná hodnota v intervalu $< 0 , 1 >$, kde dané číslo označuje věrohodnost klasifikace a zobrazeného výsledku. Čím blíže je hodnota číslu 1, tím více si je síť jistá, že správně klasifikuje.

5.2.4 Testování - Klasifikace testovacích průběhů pomocí neuronové sítě :

Po naučení sítě učební maticí byly provedeny testy správnosti klasifikace. Pro provedení klasifikace vkládaného vektoru se používá příkaz „sim“ :

```
vystup1 = sim(net2,nnvstup);
```

kde vystup1 je vektor hodnot v intervalech < 0 , 1 > , net2 je navržená a naučená síť a nnvstup je vektor hodnot, které má síť klasifikovat. Výsledný vektor hodnot může obsahovat všechny hodnoty nenulové a je nutné podmínkou získat nejvyšší číslo, které je považováno za nejpravděpodobnější podobu výsledku. Dané nejvyšší číslo má ke své pozici přiřazenou klasifikační větu a ta je následně zobrazena, jak je uvedeno výše.
Nyní byly postupně vkládány vektory blízké vektorům z učební matice. Zde dosahuje aplikace 100% úspěšnosti, kdy z deseti provedených měření byl výsledek vždy správný a v devíti případech překročil hodnotu věrohodnosti tvrzení 0,95.
Následně po přechodu k testování průběhů se stejnou diagnózou k naučeným průběhům došlo ke snížení věrohodnosti tvrzení a došlo i výjimečně ke špatné klasifikaci. Při testování jsem použil 3 různé průběhy z databáze atriových(síňových) fibrilací umístěné na serveru *physionet.org* . Průběhy prvního testování jsou zobrazeny v níže uvedené tabulce :

Tabulka naměřených hodnot během testování sítě :

Měření :	Vybraná věta :	Věrohodnost
1	**Fibrilace síní (i hrubovlnná)**	**0,7488**
2	**Fibrilace síní (i hrubovlnná)**	**0,493**
3	**Fibrilace síní (i hrubovlnná)**	**0,7879**
4	**Fibrilace síní (i hrubovlnná)**	**0,6586**
5	**Fibrilace síní (i hrubovlnná)**	**0,6073**
6	Fibrilace síní	0,5797
7	**Fibrilace síní (i hrubovlnná)**	**0,6981**
8	**Fibrilace síní (i hrubovlnná)**	**0,6821**
9	**Fibrilace síní (i hrubovlnná)**	**0,7642**
10	**Fibrilace síní (i hrubovlnná)**	**0,7836**

Tab.1 Sinove_fibrilace_TESTER.txt

Jedná se o zašuměný signál, který síť rozpozná vždy správně. Dále jsem testoval další signál s fibrilací. U něj aplikace nedosahovala vždy požadovaného výsledku a proto jsem přidal počet učení. Obě tabulky jsou uvedeny níže :

Tabulka naměřených hodnot během testování sítě :

Měření :	Vybraná věta :	Věrohodnost	1. Alternativa :	Věrohodnost
1	**Fibrilace síní**	**0,95589**		
2	Deprese ST => Angína Pectoris	0,23122	**Fibrilace síní**	**0,21162**
3	Deprese ST => Angína Pectoris	0,73945		
4	**Fibrilace síní**	**0,90683**		
5	**Fibrilace síní**	**0,70476**		
6	**Fibrilace síní**	**0,86365**	Deprese ST => Angína Pectoris	0,34824
7	Pomalé,nebo zadržené dýchání	0,79363	**Fibrilace síní**	**0,39848**
8	**Fibrilace síní**	**0,71291**	Deprese ST => Angína Pectoris	0,51887
9	**Fibrilace síní**	**0,82205**	Pomalé,nebo zadržené dýchání	0,61164
10	**Fibrilace síní**	**0,83899**		

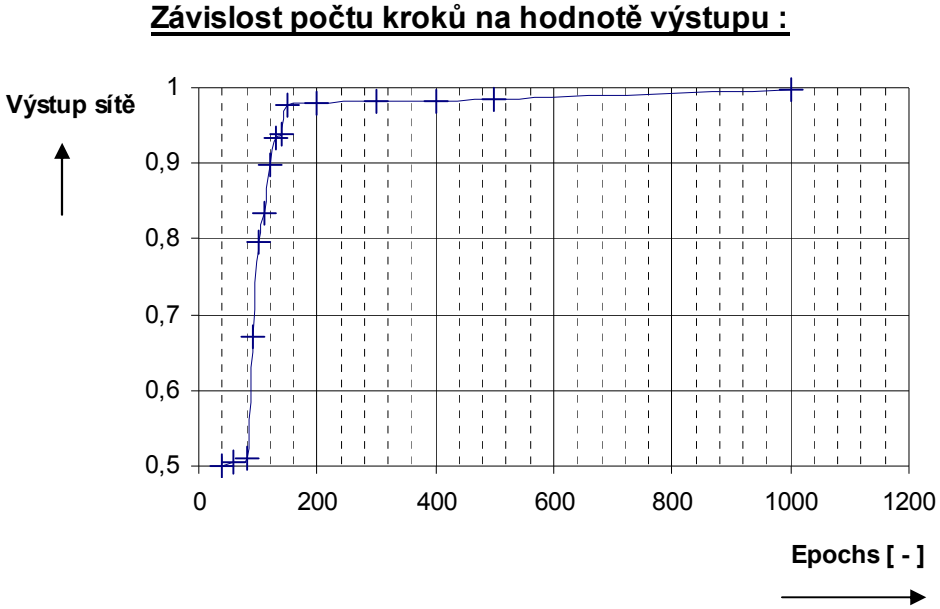
Tab.2 Sinove_fibrilace_TESTER2.txt (3x učení)

Tabulka naměřených hodnot během testování sítě :

Měření :	Vybraná věta :	Věrohodnost	1. Alternativa :	Věrohodnost
1	Deprese ST => Angína Pectoris	0,63344	**Fibrilace síní**	**0,27319**
2	**Fibrilace síní**	**0,76187**	Pomalé,nebo zadržené dýchání	0,30292
3	**Fibrilace síní**	**0.59474**		
4	**Fibrilace síní**	**0,81024**		
5	**Fibrilace síní**	**0,41242**		
6	**Fibrilace síní**	**0,67498**		
7	**Fibrilace síní**	**0,54792**		
8	**Fibrilace síní**	**0,74961**	Deprese ST => Angína Pectoris	0,47867
9	Pomalé,nebo zadržené dýchání	0,81612	**Fibrilace síní**	**0,36403**
10	**Fibrilace síní**	**0,74736**		

Tab.3 Sinove_fibrilace_TESTER2.txt (6x učení)

S dvojnásobným počtem učení ubylo chyb při klasifikaci a síť vždy nabídla fibrilaci síní alespoň v alternativní diagnóze. Došlo tedy k nepatrnému zlepšení. Pokud by se tento signál přidal do matice naučených signálů, tak by síť tento průběh rozeznala s vyšší věrohodností v každém bodě měření. Počet opakování, počet kroků a požadovaná maximální kvadratická chyba je stanovena pevně na základě experimentálního měření. Následující graf ukazuje vliv počtu kroků při učení :



Obr. 22 Graf závislostí počtu kroků na výstupu

Měření bylo provedeno na jednoduché síti, které stačilo k naučení 150 kroků. Síť v klasifikační aplikaci potřebuje průměrně 300 kroků k naučení. Je volena hodnota 2000 kroků u prvního učení a 1000 u dalších opakování učení. Učební matici lze tedy rozšiřovat, aniž by musel být nastaven vyšší počet kroků. Stejný vliv na výstup má i nastavení maximální tolerované kvadratické chyby. Čím je tato hodnota nižší, tím přesnějších lze dosáhnout výsledků. Již při hodnotě 0.5 dochází k chybám na straně sítě. Nejlepších výsledků bylo dosaženo s maximální kvadratickou chybou rovnou 0,001 .

6 Realizace grafického programu

V prostředí Matlab slouží k tvorbě grafického rozhraní aplikace guide. Každý blok, či textové pole má svůj kód uvnitř pomocného m-filu, který je pojmenovaný "Otevřít.m" Každému prvku se zvlášť musí nadefinovat funkce a zobrazované proměnné. K přenosu mezi funkcemi jednotlivých prvků nelze používat běžné proměnné, ale speciální handles pole, do kterých lze proměnné ukládat. Uvnitř funkce lze z handles dostat danou proměnnou příkazem „get" . Naopak pro vložení proměnné, či datového vektoru do handles slouží příkaz „set" . Graf se vkládá do určeného pole. To se určí a zaktivuje příkazem „axes" s názvem pole. Příkaz pro vygenerování samotného grafu má tvar :

```
plot(handles.grafcelk,timx,Signal);
```

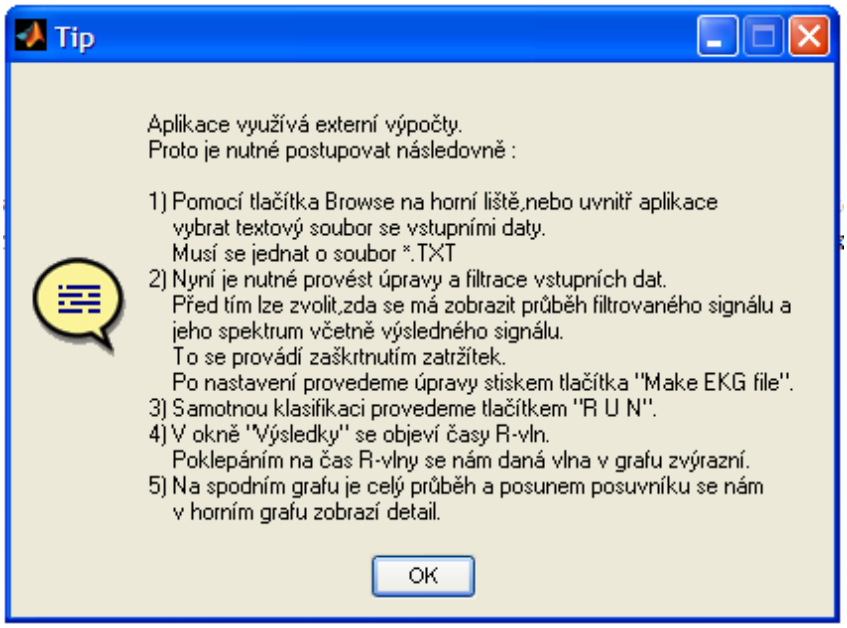
jedná se o pořadí od proměnné grafu přes osu x k ose y. Příkaz „plot" zůstává nezměněn. Volání externích m-souborů se provádí vložením funkcí externích souborů přímo do funkce daného prvku v grafickém prostředí. Po vykonání externího programu běh pokračuje v daném souboru

„Otevrit.m“ . Pro vyvolání bodů uvnitř grafu se musí v prvku ovládajícím třeba zobrazení R vln opět vygenerovat celý nový graf, nebo vložit další pozměněný graf přes původní graf.
Po seznámení s prací v guide prostředí lze vytvořit funkční aplikaci podle návrhu uvedeného v kapitole 4.1 *Finální návrh vzhledu aplikace*. Oproti návrhu je vytvořen navíc ovládací panel v horní části plochy včetně návodu k použití. Ten se vkládá v podobě chybové hlášky následovně:

```
zprava{1,1}=('Aplikace využívá externí výpočty.');
zprava{2,1}=('Proto je nutné postupovat následovně :');
zprava{4,1}=('1) Pomocí tlačítka Browse na horní liště,nebo uvnitř aplikace');
zprava{5,1}=('    vybrat textový soubor se vstupními daty.');
zprava{6,1}=('    Musí se jednat o soubor *.TXT');
zprava{7,1}=('2) Nyní je nutné provést úpravy a filtrace vstupních dat.');
zprava{8,1}=('Před tím lze zvolit,zda se má zobrazit průběh filtrovaného signálu a ');
...

msgbox(zprava,'Tip','help')
```

Příkaz msgbox vyvolá vlastní okno zobrazující proměnnou různého typu. V tomto případě zobrazí celý obsah proměnné zprava. Finální podoba návodu obsluhy je zobrazena na níže zobrazeném obrázku 23 :



Obr. 23 Grafické provedení návodu obsluhy.

Samotné oživení celé aplikace a propojení s externími výpočtovými m-soubory zabralo velkou část času určeného pro tvorbu bakalářské práce. Výsledkem je však aplikace velice podobná navržené s veškerou funkčností. Aplikace umožňuje zobrazit, či ukrýt průběhy filtrací, barevně označit R-vlny a RR intervaly vybrané uživatelem a pomocí posuvnému posouvat signálem uvnitř okna sloužícího jako lupa. V poli ovládacích prvků je zobrazen čas použitý pro běh externích m-souborů. Funkčnost aplikace je otestovaná. Její konečná podoba za běhu je zobrazena na obrázku 24 při klasifikaci signálu majícího negativní T vlny.

Obr. 24 Aplikace za běhu

Nad zobrazeným celým průběhem jsou vypsané veškeré R vlny s časy. Pod grafickým průběhem jsou od leva zobrazeny RR intervaly, S a T vlny , některé vážné nedetekované problémy a výstupní klasifikace provedená umělou neuronovou sítí. Průměrné časy pro zpracování minutového signálu se pohybují dohromady kolem 25 sekund. Zdlouhavé je učení sítě, které se opakuje při každém novém vloženém signálu. To se dá odstranit vytvořením souboru s daty naučené sítě a provádět pouze simulace na síti načtené z externího souboru. U aplikace je počítáno, že se bude jednou načítat signál přímo z přístroje. Díky tomu bude moct být odstraněno tlačítko „Browse“ a „Make EKG File“. Tím se docílí ideální jednoduchosti programu pro lékaře. Pro použití v lékařství je však nutné rozšířit databázi učební matice. V této podobě je matice velice omezená a samotná síť klasifikuje zcela neznámé signály s průměrnou věrohodností pouze kolem hodnoty 0,7 . Je však důležité, že dané problémy je systém schopen i při tak slabé učební matici rozpoznat a klasifikovat alespoň s nějakou váhou přesvědčení.

7 Závěr

Tato bakalářská práce je zaměřená na přehlednou klasifikaci EKG signálu za použití umělých neuronových sítí. Ty jsou ke klasifikaci velice vhodné a dokáží při správném použití vykazovat překvapivě dobré výsledky při klasifikaci. Pro reálnost tvorby podobné aplikace jsou nutné větší znalosti kolem srdečního rytmu a principu vzniku srdečních ozev včetně fyzikálního principu. Tím se zabývá úvodní část práce. Prostředí pro realizaci klasifikační aplikace bylo zvoleno od firmy The Mathworks s názvem Matlab. Jeho použití není snadné, ale je právem považováno za dnes nejvhodnější prostředí pro realizaci takové úlohy. S programem se nepracuje příliš snadno a proto nebyly v této práci využity možné výhody, které by urychlily a zpřesnily klasifikaci. Při tvorbě návrhu vzhledu a

rozložení tvořené aplikace bylo přihlédnuto na skutečné požadavky lékaře a představy veřejnosti o dobré aplikaci. Samotná aplikace je velice blízká ideálnímu návrhu a umožňuje velkou část operací, které byly požadovány. K úpravám signálu byly použity filtrační metody probrány v rámci bakalářského studia. Jmenovitě se jedná o FIR filtry a rychlou Fourierovu transformaci. Při volbě architektury umělé neuronové sítě nebylo přihlédnuto k doporučením a pro jednoduchost a přístupnost ze strany Matlabu byla volena dopředná dvouvrstvá perceptronová síť. Výsledná aplikace byla kompletně otestována na všech signálech z databáze Intracardiac Atrial Fibrillation Database (iafdb) a dalších náhodných signálech z jiných databází. Testování proběhlo úspěšně, ale nesplnilo zcela všechna očekávání. Testování vybraných průběhů jsou uvedeny v tabulkách.

8 Seznam použité literatury :

[1] Jiří Svršek , *Historie EKG*
[2] John R. Hampton, *EKG PRO PRAXI*, Grada , 1997
[3] John R. Hampton, *EKG stručně,jasně,přehledně*, Grada, 1996
[4] R.Schroder a H.Sudhof, *EKG v praxi*, Avicenum Praha, 1973
[5] MUDr.Lanka Borská, Ph.D. a kolektiv, *EKG DESATERO*, MSD Brno 2006
[6] Ing.František Hakl, CSc. a Ing.RNDr.Martin Holeňa CSc. , *Úvod do teorie neuronových sítí* , ČVUT 1998
[7] Doc. Ing. Jiří Kozumplík , CSc. , *Vlnkové transformace a jejich využití pro filtraci signálu EKG*, VUTIUM Brno, 2005
[8] Sandhay Samarasinge, *Neural Network for Applied Sciences and Engineering*, Auerbauch Publications NY
[9] Ing.Jan Macek, *Klasifikace EKG pomocí vlnkové transformace a rozhodovacích stromů*
[10] Mark Craven and David Page, *Hidden Markov Models*, 2006
[11] Ian H. Witten, *WEKA Machine Learning Algorithms in Java*, Morgan Kaufmann 2000
[12] Dr.Ing.David Cuesta Frau, *Dynamic Time Warping*, Polytechnic University of Valencia (Spain)
[13] Gabriela Andrejková a Peter Sinèák, *Neurónové siete Inžiniersky prístup*, Košice 1996
[14] Doc. Ing. Jana Tučková CSc. , *Úvod do problematiky umělých neuronových sítí*, ČVUT 2005
[15] Pavel Vála, *Neuronové sítě v řízení systémů*, 2002
[16] Jan Nejvárek, *MatLab a Neuronové Sítě*, FEI VUT Brno, 2002
[17] MUDr. Petr Haman, Plzeň, http://ekg.kvalitne.cz
[18] The MathWorks Inc. Matlab Help
[19] Richard Dybowski and Vanya Gant, *Clinical applications of artificial neural networks*
[20] Bc. Jan Zetek, *Zpracování biologických dat*, ČVUT 2002
[21] Bortolan, G.; Degani, R.; Willems, J.L. *Neural networks for ECG classification*, Computers in Cardiology 1990
[22] Rosaria Silipo, Carlo Marchesi, *Artificial Neural Networks for automatic ECG analysis*, (1998)

9 Seznam použitých symbolů :

a.…...výstup perceptronu
b.……prahová hodnota
f ……aktivační funkce
N …..počet koeficientů/řád filtru
p ….. počet vstupů
w …..váhy spoje
x ….. vstupy
X …..vektor spektra frekvence
y ….. výstupy
Y …..výsledný vektor „četnosti frekvence“
ω_N…..úhlová frekvence

10 Seznam použitých zkratek :

CWT	Spojitá vlnková transformace
DNA	Genetická informace
DTW	Dynamic Time Warping
ECG	Elektrokardiogram
EKG	Elektrokardiogram
FFT	Rychlá Fourierova transformace
FIR	Finite impulse response
GUIDE	Graphical User Interface development environment
HMM	Hidden Markov Models
iFFT	Inverzní rychlá Fourierova transformace
LabView	Laboratory Virtual Instrumentation Engineering Workbench
QRS	Sestava tří vln označených Q,R a S
ss	Stejnosměrný

11 Seznam použitých zkratek :

1. Výpis souboru EKG_open.m
2. Výpis souboru QRS_test.m
3. Výpis souboru Otevrit.m

12 Přílohy

12.1 Soubor EKG_open.m

```
function [File_Name,caszprac,filtrgraf,specgraf] =
EKG_open(File_Name,caszprac,filtrgraf,specgraf)
clc;
%**************************************************************************
% Vstupní data
%**************************************************************************
%File_Name = input('Napiš název souboru matice EKG signálu = ','s'); % *.TXT
EKG_Data = load(File_Name) ;                % Načtení
t_CPU=cputime;
Cas = EKG_Data(6:end,1);                    % Rozdělení dat na čas
n = length(EKG_Data( 6 ,:))
grafu=n-1;
konecny=1;
ulozif=0;
disp(['Délka vektoru EKG : '...
    ,num2str(length(EKG_Data( : ,1)),'%10.0f'),' vzorků']);
fvzv = round(1/(Cas(10,1)-(Cas(9,1))));
fvzx=0;
 if (fvzv) < 300
       fvzx=0+300;
 end
 fvz=fvzx+fvzv
for  i= 2 : n
    EKG(:,i) = EKG_Data(:,i);          % Druhý až n-tý sloupec=data

    Rvrchol = max(EKG(:,i))

if grafu<6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%==========================Pro méně,než 6 průběhů=========================%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    YY=length(EKG(6:2:end,i));           %%
    EKGsmall(1:1:YY,i) = EKG(6:2:end,i);%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-----------------Úpravy a kontrola délek vektorů--------------------------

    Delka=length(EKGsmall(:,i));
    YYt=length(Cas(1:2:end));           %%
    t1(1:1:YYt) = Cas(1:2:end);%%
    if Delka<1000
        YY=length(EKG(6:1:end,i));
        EKGsmall(1:1:YY,i)= EKG(6:1:end,i);
        t1=Cas;
    end

    Delka=length(EKGsmall(:,i));
    if length(t1)~=Delka
        disp(['Délka 1. vektoru času '...
        ,num2str(length(t1),'%10.1f'),'sec a délka druhého vektoru dat :'...
        ,num2str(Delka,'%10.1f')]);
    end
disp('******************************************************');
disp('Nutná kontrola pravého a určeného času:');
disp(['Konečký čas : '...
     ,num2str(Cas(end),'%10.1f'),'sec a spočtený konečný čas :'...
     ,num2str(t1(end),'%10.1f')]);
disp('******************************************************');

%-----------------Zobrazení vstupních dat v grafu--------------------------
% figure(1);
% set(figure(1),'NumberTitle','off');
% set(figure(1),'Name','Vstupní signál EKG');
%    subplot(grafu,1,i-1)
%     plot(t1,EKGsmall(:,i),'r');
%     hold on;
%     %================================
%         X_Min = t1(1);
%         X_Max = t1(Delka-1);
```

```
%         Axis_Y = ylim;
%         Y_Min = Axis_Y(1);
%         Y_Max = Axis_Y(2);
%         axis([X_Min X_Max/3 Y_Min Y_Max]);
%             if Delka<1000
%                 axis([X_Min X_Max Y_Min Y_Max]);
%             end
%          Plot_Title = strcat('EKG_', i+47 , ' ze souboru : ', File_Name);
%         Plot_Title = regexprep(Plot_Title, '_', '-');
%         title(Plot_Title);
%         xlabel('t [sec]');
%         ylabel('A [mV]');
%         %================================

%-----------------Hledání maxima a grafické zobrazení----------------------
% index=find(EKGsmall(:,i)==max(EKGsmall(:,i)));
% Perioda=num2str(t1(index));
% maxamp=num2str(max(EKGsmall(:,i)));
% plot(t1(index),max(EKGsmall(index-5:index+5,i)),'g.', 'MarkerSize',25);
% text(t1(index)+0.2,EKGsmall(index),['Perioda = ',Perioda]);
% text(t1(index)+0.2,EKGsmall(index)+0.5,['Amplituda = ',maxamp]);

hold off;
    %>>>>>>>>>>>>>>>>>>>>>>>>>FILTRACE<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
    %>>>>>>>>>>>>>>>>>>>>>>>>>========<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


    D = fir1(20, [0.168 0.260], 'stop');
    C = fir1(10, [0.001 0.002], 'stop');
    B = fir1(70, [0.150 0.999] , 'stop');
    filtekg(:,i) = filter(B,1,EKG(:,i));
    filtekga(:,i) = filter(C,1,filtekg(:,i));
    filtekga(:,i) = filter(C,1,filtekga(:,i));
    filtekgb(:,i) = filter(D,1,filtekga(:,i));
    filtekgb(:,i) = filter(D,1,filtekgb(:,i));
    filtekgb(:,i) = filter(D,1,filtekgb(:,i));

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 zpozdeni=84;                                                      %%
 YYY=length(filtekgb(zpozdeni:2:end,i));                           %%
 EKGsmallfilt(1:YYY,i) = filtekgb(zpozdeni:2:end,i);               %%
 YYYt=length(Cas(1:2:end-(zpozdeni-6)))                            %%
 tf(1:1:YYYt) = Cas(1:2:end-(zpozdeni-6));                         %%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    ktest=length(EKGsmallfilt(:,i));

if ktest<1000     %pro malý počet vzorků jiná filtrace
        EKGsmallfilt(:,i)= zeros(1);
        BB = fir1(20, [0.168 0.232], 'stop');
        AA = fir1(10, [0.300 0.999] , 'stop');
        filtekg(:,i) = filter(AA,1,EKG(:,i));
        filtekg(:,i) = filter(BB,1,filtekg(:,i));
        %EKGsmallfilt= filtekg;

        pozdejo=20;
        YYYY=length(filtekg(pozdejo:end,i));
        EKGsmallfilt(1:YYYY,i) = filtekg(pozdejo:end,i);
        YYYt=length(Cas(1:1:end-(pozdejo-6)))
        tf(1:1:YYYt) = Cas(1:1:end-(pozdejo-6));
end

    k=length(EKGsmallfilt(:,i))

%----------Srovnání postupného filtrování a časů vektorů-------------------
%     krok=(1/fvz)*nasob
%     tf=0:krok:((k-1)/fvz)*nasob;

%                 figure(4);
%                 plot(tf,filtekg(1:length(tf-1),i),'b');
%                 hold on;
%                  plot(tf,filtekga(1:length(tf-1),i),'g');
%
%                 plot(tf,filtekgb(1:length(tf-1),i),'r');
%                  hold off;
disp('*****************************************************');
disp('Nutná kontrola pravého a určeného času:');
disp(['Konečký čas : '...
        ,num2str(Cas(end),'%10.1f'),'sec a spočtený konečný čas :'...
        ,num2str(tf(end),'%10.1f')]);
disp('*****************************************************');
```

```
%-----------------Při zatržení zatržítka se zobrazí graf-------------------
if filtrgraf==1

    figure(2);
    set(figure(2),'NumberTitle','off');
    set(figure(2),'Name','Filtrovaný EKG');
    subplot(grafu,1,i-1);
    plot(tf,EKGsmallfilt(:,i),'r');
    hold on;
    %plot(tf,EKGsmall(1:length(tf-1),i),'g');   % pro srovnani
    %================================
         X_Min = tf(1);
         X_Max = tf(k-1);
         Axis_Y = ylim;
         Y_Min = Axis_Y(1);
         Y_Max = Axis_Y(2);
         axis([X_Min X_Max/3 Y_Min Y_Max]);
          if k<1000
          axis([X_Min X_Max Y_Min Y_Max]);
         end
        Plot_Title = strcat('EKG_', i+47 , ' filtrované : ');
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%************************************************************************
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>Spektra<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%************************************************************************

   M = length(EKGsmallfilt(:,i));
   f= -fvz/2 : fvz/M : ((M-1)*fvz/(2*M));
   spek=fft(EKGsmallfilt(:,i));
   velspek=length(spek)-1
   spek(1:9)=zeros(1);
   spek(end-6:end)=zeros(1);
    if velspek>2000
        spek(1:15)=zeros(1);
        spek(end-10:end)=zeros(1);
%    spek(800:1200)=zeros(1);
%    spek(end-1200:end-800)=zeros(1);
    end
    if velspek>7000
        spek(1:50)=zeros(1);
        spek(end-40:end)=zeros(1);
    end
   signal_nonss=ifft(spek);  % inverzni fft

%-----------------Při zatržení zatržítka se zobrazí graf-------------------
if specgraf==1
figure(3);
    set(figure(3),'NumberTitle','off');
    set(figure(3),'Name','FFT a spec EKG');

   subplot(grafu+1,2,i-1);
    plot(f,abs(fftshift(fft(signal_nonss))));
    title('Spektrum signálu EKG po filtraci a bez ss složky');
    xlabel('frekvence [Hz]');
    ylabel('Amplituda');
end
Ltest=length(signal_nonss);
        nasob=2;
        if ktest<1000
            nasob=4;
        end
    krok=(1/fvz)*nasob;
    YYYYYt=length(Cas(1:1:end));
    %%t1(1:1:YYYYYt) = Cas(1:2:end);%%
    tf=0:krok:((Ltest-1)/fvz)*nasob;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if specgraf==1
    subplot(grafu+1,2,i+1);
    plot(tf,real(signal_nonss));
    %================================
         X_Min = tf(1);
         X_Max = tf(k-1);
         Axis_Y = ylim;
         Y_Min = Axis_Y(1);
         Y_Max = Axis_Y(2);
```

```
         axis([X_Min X_Max/3 Y_Min Y_Max]);
         if k<1000
         axis([X_Min X_Max Y_Min Y_Max]);
        end
        Plot_Title = strcat('EKG_', i+47 , ' po ifft bez SS složky: ');
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
        hold on;
     plot(tf,EKGsmallfilt(:,i),'r');
     %index=find(EKGsmallfilt(:,i)==max(EKGsmallfilt(:,i)));
     %mainPer=num2str(tf(index));
     %mainmaxA=num2str(max(EKGsmallfilt(:,i)));
end
% plot(tf(index),max(EKGsmallfilt(index-5:index+5,i)),'g.', 'MarkerSize',25);
%      text(tf(index)+0.2,EKGsmallfilt(index),['Perioda = ',mainPer]);
%      text(tf(index)+0.2,EKGsmallfilt(index)+0.2,['Amplituda = ',mainmaxA]);

hold off;
anone=0;
for poradi=1:length(signal_nonss);
    if signal_nonss(poradi) > 0.55
        anone=1 ;
    end
end
    if anone==1
        disp('*********************************************************');
        disp(['Zapisuji do souboru koncovky EKG  '...
    ,num2str(i-1,'%10.0f'),'.  zobrazenou křivku.']);
        disp('*********************************************************');
        konecekg(:,konecny)=signal_nonss;
        konecny=konecny+1;
        ulozif=ulozif+1;
    end


  elseif grafu<11
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%==========================Pro méně,než 11 průběhů=========================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    EKGsmall(:,i) = EKG(13:2:end,i);%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Delka=length(EKGsmall(:,i));
    nasob=2;
    if Delka<1000
        EKGsmall= EKG;
        nasob=4;

    end
    Delka=length(EKGsmall(:,i));
    krok=(1/fvz)*nasob;
    t1=0:krok:((Delka-1)/fvz)*nasob;
if filtrgraf==1
figure(1);
set(figure(1),'Name','Vstupní signál EKG');
    subplot(grafu/2,2,i-1)
    plot(t1,EKGsmall(:,i),'r');
    hold on;
    %================================
        X_Min = t1(1);
        X_Max = t1(Delka-1);
        Axis_Y = ylim;
        Y_Min = Axis_Y(1);
        Y_Max = Axis_Y(2);
        axis([X_Min X_Max/3 Y_Min Y_Max]);
        if Delka<1000
         axis([X_Min X_Max Y_Min Y_Max]);
        end
        Plot_Title = strcat('EKG_', i+47 , ' ze souboru : ', File_Name);
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
end
    %>>>>>>>>>>>>>>>>>>>>>>>>FILTRACE<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
    %>>>>>>>>>>>>>>>>>>>>>>>>========<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
            %     D = fir1(50, [0.150 0.250], 'stop');
            %     C = fir1(50, [0.002 0.02], 'stop');
            %     B = fir1(50, [0.255 0.99] , 'stop');
            %     filtekg(:,i) = filter(B,1,EKG(:,i));
            %     filtekga(:,i) = filter(C,1,filtekg(:,i));
            %     filtekgb(:,i) = filter(D,1,filtekga(:,i));
    D = fir1(20, [0.168 0.260], 'stop');
    C = fir1(19, [0.001 0.02], 'stop');
    B = fir1(70, [0.150 0.999] , 'stop');
    filtekg(:,i) = filter(B,1,EKG(:,i));
    filtekga(:,i) = filter(C,1,filtekg(:,i));
    filtekga(:,i) = filter(C,1,filtekga(:,i));
    filtekgb(:,i) = filter(D,1,filtekga(:,i));
    filtekgb(:,i) = filter(D,1,filtekgb(:,i));
    filtekgb(:,i) = filter(D,1,filtekgb(:,i));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    EKGsmallfilt(:,i) = filtekgb(60:2:end,i);%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ktest=length(EKGsmallfilt(:,i));
        nasob=2;
    if ktest<1000
        EKGsmallfilt= filtekg;
        nasob=3;

    end
    k=length(EKGsmallfilt(:,i));
    krok=(1/fvz)*nasob;
    tf=0:krok:((k-1)/fvz)*nasob;


if filtrgraf==1

    figure(2);
    set(figure(2),'NumberTitle','off');
    set(figure(2),'Name','Filtrovaný EKG');
    subplot(grafu/2,2,i-1);
    plot(tf,EKGsmallfilt(:,i),'r');
    hold on;
    %plot(tf,EKGsmall(1:length(tf-1),i),'g');
     %================================
         X_Min = tf(1);
         X_Max = tf(k-1);
         Axis_Y = ylim;
         Y_Min = Axis_Y(1);
         Y_Max = Axis_Y(2);
         axis([X_Min X_Max/3 Y_Min Y_Max]);
         if k<1000
         axis([X_Min X_Max Y_Min Y_Max]);
        end
        Plot_Title = strcat('EKG_', i+47 , ' filtrované : ');
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
end
%==========================================================================
%**************************************************************************
%>>>>>>>>>>>>>>>Spektra<<<<<<<<<<
%**************************************************************************

    M = length(EKGsmallfilt(:,i));
   f= -fvz/2 : fvz/M : ((M-1)*fvz/(2*M));
   spek=fft(EKGsmallfilt(:,i));
   spek(1)=zeros(1);
   spek(800:end)=zeros(1);

 signal_nonss=ifft(spek);                    % inverzni fft
if specgraf==1
 figure(3);
    set(figure(3),'NumberTitle','off');
    set(figure(3),'Name','FFT a spec EKG');
    subplot(grafu/2,2,i-1);

        plot(f,abs(fftshift(fft(signal_nonss))));
        Plot_Title = strcat('Spektrum EKG_', i+47 ,' po filtraci bez SS složky: ');
        title(Plot_Title);
        xlabel('f [Hz]');
        ylabel('Amplituda ');
        Ltest=length(signal_nonss);
        nasob=2;
        krok=(1/fvz)*nasob;
```

```
        tf=0:krok:((Ltest-1)/fvz)*nasob;
figure(4);

    set(figure(4),'NumberTitle','off');
    set(figure(4),'Name','Konečný signál');
    subplot(grafu/2,2,i-1);

        plot(tf,real(signal_nonss));
        %================================
         X_Min = tf(1);
         X_Max = tf(Ltest-1);
         Axis_Y = ylim;
         Y_Min = Axis_Y(1);
         Y_Max = Axis_Y(2);
         axis([X_Min X_Max/3 Y_Min Y_Max]);
        if Ltest<1000
            axis([X_Min X_Max Y_Min Y_Max]);
        end
        Plot_Title = strcat('EKG_', i+47 , ' po ifft bez SS složky: ');
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
        hold on;
    plot(tf,EKGsmallfilt(:,i),'r');
end
anone=0;
for poradi=1:length(signal_nonss);
    if signal_nonss(poradi) > 0.6
        anone=1 ;
    end
end
    if anone==1
        disp('******************************************************');
        disp(['Zapisuji do souboru koncovky EKG  '...
    ,num2str(i-1,'%10.0f'),'.  zobrazenou křivku.']);
        disp('******************************************************');
        konecekg(:,konecny)=signal_nonss;
        konecny=konecny+1;
        ulozif=ulozif+1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%==========================Pro více,než 10 průběhů=========================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

else
   disp('Průběhů je více, jak 10. Někde je chyba.');
   msgbox('Průběhů je v souboru příliš mnoho a zpracování by bylo příliš
zdlouhavé','Tip','error')
end
end

%*************************************************************************
% Uložení dat do souboru
%*************************************************************************
File_Name = regexprep(File_Name, '.txt','.ekg');
if ulozif ~= 0
 for likvid=2:1:length(tf)
     if tf(likvid-1)< tf(likvid)
         endtime(likvid,1)=tf(likvid);
       konecekgfine(likvid,:)=konecekg(likvid,:);
     else
         likvid=length(tf)-1;
     end
 end

save(strcat(File_Name,'',''),'endtime' ,'konecekgfine');
msgbox('Nyní po stisku tlačítka RUN dojde ke klasifikaci.','Tip','help')
else
msgbox('Pro nepoužitelnost vkládaného signálu nebyl vytvořen soubor EKG.','Tip','error')
end
%**************************
TimeC=cputime-t_CPU;
disp(['Čas potřebný pro výpočet: ',num2str(TimeC, '%10.2f'), ' s']);
disp('==========================================');
[caszprac]=(['Čas potřebný pro výpočet: ',num2str(TimeC, '%10.2f'), ' s']);
```

12.2 Soubor QRS_test.m

```
function [File_Name,prenosna,prenosna2,caszprac,timx,Signal,puk,bod,RRinterval,probdet,nnet]...
    =QRS_test(File_Name,prenosna,prenosna2,caszprac,timx,Signal,puk,bod,RRinterval,probdet,nnet)
clc; %clear all; close all;
%*************************************************************************
% Vstupní data
%*************************************************************************
t=cputime;

%parametry pro ANN síť
aaa=0.000;
bbb=0.000;
ccc=0.000;
ddd=0.000;
eee=0.000;
fff=0.000;
ggg=0.000;
hhh=0.000;
chchch=0;
k=1;
kk=1;
kkk=1;
kkkk=1;
k5=1;
k6=1;
k7=1;
k8=1;
problem=0;
%File_Name = input...
%    ('Napiš název souboru matice EKG signálu s koncovkou ekg = ','s');
load ('-mat',File_Name);
Signal=real(konecekgfine(:,1));
timx=real(endtime(:,1));
indexovani=1;
indexprob=2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Derivace                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Uroven=0.3*max(Signal(:))

for i=1:length(Signal(:))
   if Signal(i)>=0
      klad(i)=Signal(i);
   else
      klad(i)=-(Signal(i));
   end
end
    % figure(3);
    % plot(klad)
    for i=1:length(Signal(:))
       if klad(i)>=Uroven
          vecder(i)=klad(i);
       else
         vecder(i)=Uroven;
       end
    end

for i=2:length(Signal(:))-1
   y(i)=vecder(i+1)-vecder(i-1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%figure(1);
   %plot(timx(1:length(y)),y(:));
   hold on;
   %================================
        X_Min = timx(1);
        X_Max = timx(length(y)-1);
        Axis_Y = ylim;
        Y_Min = Axis_Y(1);
        Y_Max = Axis_Y(2);
        axis([X_Min X_Max Y_Min Y_Max]);
        Plot_Title = strcat('EKG_', 1+48 , ' ze souboru : ', File_Name);
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Podmínka amplitudy pro získání správného počtu R vln                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=0;
i=1;
index=1;
while i < (length(Signal)-3)
   if y(i)>0.55*max(y(:))
        n=n+1;
        puk(index)= i;
        index=index+1;
        i=i+80;
   else
      i=i+1;
   end
end
zoom on;
disp('================================================');
disp(['celkem jsem našel: ' num2str(n) ' QRS komplexů.'])
[pomoci{1,1}]=(['QRS komplexů celkem : ' num2str(n) ]);

% figure(2);
% plot(timx(1:length(Signal)),Signal);
% hold on;
% %================================
%         X_Min = timx(1);
%         X_Max = timx(length(Signal)-1);
%         Axis_Y = ylim;
%         Y_Min = Axis_Y(1);
%         Y_Max = Axis_Y(2);
%         axis([X_Min X_Max Y_Min Y_Max]);
%         Plot_Title = strcat('EKG_', 2+47 , ' ze souboru : ', File_Name);
%         Plot_Title = regexprep(Plot_Title, '_', '-');
%         title(Plot_Title);
%         xlabel('t [sec]');
%         ylabel('A [mV]');

%=========================================================================
%Zobrazení R vln do EKG
%=========================================================================

 disp('================================================');
for bod=1:length(puk)

%plot(timx(puk(bod)),max(Signal(puk(bod)-5:puk(bod)+5)),'g.', 'MarkerSize',25);
tRvlna(bod)=timx(puk(bod));%*((timx(3)-timx(2)))
if (puk(bod)<length(Signal(:))-40)
%=========================================================================
%Detekce chyb v R vlnách a zápis R vln do aplikace
%=========================================================================
    if Signal(puk(bod)+22)> 0.5*max(y(:))
    disp(['Detekce infarktu na ',num2str(bod),'. R vlně. '])
    eee=eee+0.5;
        if Signal(puk(bod)+30)> 0.5*max(y(:))
            eee=eee+1;
        end
    end
end
%disp(['Čas ',num2str(bod),'. R vlny : ', num2str(tRvlna(bod), '%10.4f'),' sec']);
pomoci{bod+2,1}=(['Čas ',num2str(bod),'. R vlny : ', num2str(tRvlna(bod), '%10.4f'),' sec']);

end


prenosna=strcat(pomoci);
%=========================================================================
%=========================================================================
%=========================================================================
%=========================================================================
%Určení RR intervalu a detekce chyb v RR intervalech
%=========================================================================
for i=2:length(tRvlna(:))
    RR(i-1)=tRvlna(i)-tRvlna(i-1);
    %disp([num2str(i-1),'. RR interval je  : ', num2str(RR(i-1), '%10.4f'),' sec']);
    RRintervaly{i-1,1}=([num2str(i-1),'. RR interval je  : ', num2str(RR(i-1), '%10.4f'),'
sec']);


end
RRinterval=strcat(RRintervaly);
    AVRR=sum(RR)/length(RR)
if AVRR < 0.6
```

```
    ddd=0.6-AVRR;
elseif AVRR > 0.9
            ddd=0.9+AVRR+1;

        else
        ddd=1;
end


    for i=2:length(RR(:))
        if RR(i)<AVRR-0.08
            problem=problem+1;
            if RR(i-1)>AVRR-0.08
                aaa=aaa+RR(i);
                k=k+1;
                probdeta{indexprob,1}=([num2str(i),'.zkrácený RR : ', num2str(RR(i)),' sec.']);
                indexprob=indexprob+1;
            end
        end
        if RR(i)>AVRR+0.08
            problem=problem+1;
            if RR(i-1)<AVRR+0.08
            bbb=bbb+(RR(i)-AVRR);
            kk=kk+1;
            probdeta{indexprob,1}=([num2str(i),'.protažený RR : ', num2str(RR(i)),' sec.']);
            indexprob=indexprob+1;
            end
        end

        if RR(i)>(2*AVRR-0.2)
            ccc=ccc+(RR(i)-AVRR);
            kkk=kkk+1;
            disp(['Detekce dvojnásobného RR v ',num2str(i),'. RR intervalu '])

        end
    end

aaa=(aaa/k)*10;
bbb=(bbb/kk)*10;
ccc=(ccc/kkk)*10;
%=========================================================================
%Určení vln T
%=========================================================================
for bod=2:length(puk)
    if timx(puk(bod)-40)-timx(puk(bod-1)+20)<(2*AVRR-0.2)
    Tvlna(bod-1)=max(Signal(puk(bod-1)+20:puk(bod)-40));
    for j=(puk(bod-1)):1:(puk(bod))
        if Tvlna(bod-1)==Signal(j)
            tTvlna(bod-1)=timx(j);
            %disp(['Čas ',num2str(bod-1),'. T vlny : ', num2str(tTvlna(bod-1), '%10.4f'),'
sec.']);
        pomoci2{indexovani,1}=(['Čas ',num2str(bod-1),'. T vlny : ', num2str(tTvlna(bod-1),
'%10.4f'),' sec',...
                ' Hodnota : ',num2str(Tvlna(bod-1),'%10.4f'),'mV.']);
        indexovani=indexovani+1;
        end
    end
    end
end
%=========================================================================
%Určení vln S a detekce negativnich T vln
%=========================================================================
for bod=1:length (puk)-1
            Svlna(bod)=min(Signal(puk(bod):puk(bod)+30));
            testSvlna(bod)=min(Signal(puk(bod):puk(bod+1)-40));
            if testSvlna(bod) ~= Svlna(bod)
                moznaT(k7)=testSvlna(bod);

                if moznaT(k7)< -0.2
                for j=(puk(bod)):1:(puk(bod+1))
                    if moznaT(k7)==Signal(j)
                        tmoznaT(k7)=timx(j);
                        if Signal(j-20)> moznaT(k7)+0.2
                            if Signal(j+20)> moznaT(k7)+0.2
                                chchch=chchch+(2-(Signal(j)));
                                probdeta{indexprob,1}=(['Detekce možné neg. T-vlny ',
num2str(moznaT(k7)),'mV v čase ',num2str(tmoznaT(k7)),' sec.']);
                                indexprob=indexprob+1;
                                k8=k8+1;
                            end
                        end
```

```
                    end
                end
                end
                eee=eee+(testSvlna(bod)-Svlna(bod));
                probdeta{indexprob,1}=(['Špatná detekce vlny S u ', num2str(bod),'. R vlny.']);
                indexprob=indexprob+1;
                k7=k7+1;
             end
    for jj=(puk(bod):puk(bod)+20)
        if Svlna(bod)==Signal(jj)
           tSvlna(bod)=timx(jj);
           pomoci2{indexovani,1}=(['Čas ',num2str(bod),'. S vlny : ', num2str(tSvlna(bod),
'%10.4f'),...
                    ' sec.',' Hodnota : ' , num2str(Svlna(bod),'%10.4f'),'mV.']);
           indexovani=indexovani+1;
           if Signal((jj+10))<(Svlna(bod)+0.2)
                    fff=fff+(Svlna(bod)+0.2)-Signal((jj+10));
                    kkkk=kkkk+1;

           end
           if Svlna(bod)< Signal(jj+20)-0.2
                    hhh=hhh+Svlna(bod)*(-2);
                    k6=k6+1;
           end
           if Svlna(bod)> -0.05
              ggg=ggg+(10*Svlna(bod));
              k5=k5+1;
              probdeta{indexprob,1}=(['Vlna S chybí u ', num2str(bod),'. R vlny.']);
              indexprob=indexprob+1;
           end
        end

    end

end
% for i=1:length(tTvlna)
%     if tTvlna(i)>tSvlna(i)
%         TS(i)=tTvlna(i)-tSvlna(i)
%     end
% end
eee=(eee/k7)*10
fff=(fff/kkkk)*10;
ggg=(ggg/k5)*10;
hhh=(hhh/k6)*10;
if k8>3
chchch=chchch/k8;
end
if chchch~=0
    if k8<4
         chchch=1;
    end
end
prenosna2=strcat(pomoci2);

  %disp('================================================');
  %disp(['Nadetekovaných problémů RR intervalu : ', num2str(problem)]);
  probdeta{1,1}=(['Nadetekovaných problémů RR intervalu : ', num2str(problem)]);
  probdet=strcat(probdeta);
% %*************************************************************************
Time=cputime-t;
% disp(['Čas potřebný pro výpočet: ',num2str(Time, '%10.2f'), ' s']);
% disp('================================================');
% disp('================================================');
[caszprac]=(['Čas potřebný pro výpočet: ',num2str(Time, '%10.2f'), ' s']);

%**************************************************************************
% Vektor pro neuronovou síť
%**************************************************************************
% aaa...RR interval příliš krátký - odchylka
% bbb...RR interval příliš dlouhý - odchylka
% ccc...RR interval poblíš dvojnásobku
% ddd...zrychlený tep-zátěž a podobně
% eee...Rvlna dlouho v aplitudě
% fff...S vlna dlouhá
% ggg...Absence S vlny
% hhh...S vlna příliš negativní
% chchch...Negativni T vlna

% nnvstup=[ a b c d e f ];
%nnvstup=[ 0 ; 0 ; 0 ; 0 ; 0 ; 1 ];
```

```
ze=0.0001;
%=========================================================================
% Učící matice
%=========================================================================

ma=[     ze ;    1.9 ;    1.8 ;   0.25 ;     -4 ;    1.5 ;      0 ;      3 ;    1.6 ];
mb=[     ze ; 0.0007 ;     ze ; 1.0000 ;     ze ;    1.4 ;     ze ;    1.6 ;     ze ];
mc=[     ze ;    0.6 ;     ze ;    0.2 ;  -0.17 ;   1.14 ;     ze ;    1.4 ;     ze ];
md=[    2.8 ;   1.37 ;     ze ;    0.1 ;-0.2366 ;   1.21 ;     ze ;    2.3 ;     ze ];
me=[    5.5;    3.6 ;     ze ; 1.0000 ; -0.9349;     ze ;     ze ;     8  ;     ze ];
mf=[     ze ;     ze ;     ze ; 1.0000 ;     ze ;     ze ;     ze ;    6.5 ;     ze ];
mg=[     ze ;     ze ;     ze ; 1.0000 ;     ze ;     ze ;     ze ;16.0570 ;     ze ];
mh=[     ze ;    0.4 ;     ze ;      3 ;     ze ;     ze ;     ze ;    8.6 ;     ze ];
mi=[      4 ;   1.13 ;     ze ; 1.0000 ;     ze ;    0.5 ;     ze ;    4.5 ;     ze ];
mj=[    6.8 ;    6.7 ;    6.5 ;    2.8 ;     ze ;     ze ;     ze ;    9.7 ;     ze ];
ml=[     ze ;    2.6 ;      3 ; 1.0000 ; -0.016 ;     ze ;     ze ;   11.8 ;     ze ];
mm=[    6.5 ;     10 ;    9.6 ; 1.8000 ; -1.5   ;   1.3  ;     ze ;     ze ;    2.5 ];
mo=[    8.1 ;    1.0 ;     ze ;   2.91 ; -2.6   ;    1.6 ;    4.1 ;     ze ;    2.3 ];
mp=[    4.3 ;    0.8 ;     ze ;    2.9 ; -0.11  ;  1.009 ;     ze ;    3.2 ; 1.9000 ];
mq=[    3.9 ;    0.6 ;      0 ; 1.0010 ;   -0.3 ;   1.18 ;      0 ;      4 ; 1.0000 ];

na=[     ze ; 1.8759 ; 1.8759 ; 0.2225 ; -4.9278 ; 1.4311 ; 0.0001 ; 2.9153 ;   1.2 ];
nb=[     ze ;     ze ;     ze ; 1.0000 ;     ze ; 1.4713 ;     ze ; 1.5998 ;     ze ];
nc=[     ze ; 0.5861 ;     ze ; 0.1102 ; -0.1795 ; 1.1606 ;     ze ; 1.4689 ;     ze ];
nd=[ 2.8706 ; 1.0039 ;     ze ; 0.0612 ; -0.0366 ; 1.2038 ;     ze ; 2.1945 ;     ze ];
ne=[ 5.5200 ; 1.5800 ; 0.0001 ; 1.0000 ; 1.2340 ;     ze ;     ze ;     ze ;     ze ];
nf=[     ze ;     ze ;     ze ; 1.0000 ;     ze ;     ze ;     ze ; 6.7808 ;     ze ];
ng=[     ze ;     ze ;     ze ; 1.0000 ;     ze ;     ze ;     ze ;16.0570 ;     ze ];
nh=[     ze ; 0.4344 ;     ze ; 2.9543 ;     ze ;     ze ;     ze ; 8.6917 ;     ze ];
ni=[ 4.1250 ; 1.1305 ;     ze ; 1.0000 ;     ze ; 0.4167 ;     ze ; 4.6271 ;     ze ];
nj=[ 6.6600 ; 6.6251 ; 6.5460 ; 2.8699 ;     ze ;     ze ;     ze ; 9.7025 ;     ze ];
nk=[     ze ;     ze ;     ze ; 1.0000 ;     ze ;     ze ;     ze ; 16.056 ;     ze ];
nl=[     ze ; 2.5073 ; 2.5073 ; 1.0000 ; -0.0157 ;     ze ;     ze ; 11.7002 ;    1 ];
nm=[  6.625 ;  9.558 ;  9.558 ; 2.0000 ; -1.0575 ; 1.398  ;     ze ;     ze ; 2.6047 ];
no=[ 8.1627 ; 1.0493 ;     ze ; 2.9021 ; -2.6502 ; 1.5767 ;  0.21 ;     ze ; 2.2001 ];
np=[ 8.2122 ; 0.9693 ;     ze ; 2.8935 ; -0.1173 ; 0.9571 ;     0 ; 3.2725 ; 1.8000 ];
nq=[ 4.0841 ; 0.6885 ;      0 ; 1.0000 ; -0.2359 ; 1.1971 ;     0 ; 4.0737 ; 1.0000 ];
vysledek{1,1}=('Nebyl detekován žádný problém');


nnvstuplearn2=[ ma , mb , mc , md , me , mf , mg , mh , mi , mj , ml , mm , mo , mp ,mq ];
nnvstuplearn =[ na , nb , nc , nd , ne , nf , ng , nh , ni , nj , nl , nm , no , np ,nq ];
nna=[ aaa ; bbb ; ccc ; ddd ; eee ; fff ; ggg ; hhh ; chchch ];
nnvstup=(nna)
[siza,sizb]=size(nnvstuplearn);
nnvystup(:,:)=zeros;
for i=1:sizb
    nnvystup(i,i)=1;
end


%=========================================================================
% Nadefinování sítě
%=========================================================================
S1 = 9;
[S2,Q] = size(nnvystup);
P =nnvstuplearn;
net1 = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
net1 = init(net1);
net1.LW{2,1} = net1.LW{2,1}*0.01;
net1.b{2} = net1.b{2}*0.01;

%=========================================================================
% 1.Trénování sítě
%=========================================================================
P = nnvstuplearn;
T = nnvystup;
net1.performFcn = 'sse';
net1.trainParam.goal = 0.6;
net1.trainParam.show = 50;
net1.trainParam.epochs = 3000;
net1.trainParam.mc = 0.95;

[net2,tr] = train(net1,P,T);
vystup1 = sim(net2,nnvstup);


ndd=[5.1 ; 4.9 ; 5 ; 0.979 ; -0.04 ; 1.5 ; 0.01 ; 5.6 ; 1.019 ];%jemnovlnná fibrilace
noo=[7.1111 ; 0.7728 ;  0 ; 2.9021; -2.6502;  1.5767;   0 ;  0 ; 2.4231 ]; %IM
nee=[ 7.5200 ; 2.1580 ; 5.0001 ; 3.0000 ; -1.9 ;     ze ;     ze ;     ze ;     ze ];%af
```

```
nnvstuplearn3=[ na , nb , nc , ndd , nee , nf , ng , nh , ni , nj , nl , nm , noo , np , nq ];
%=========================================================================
% 2.Trénování sítě
%=========================================================================
P2 =nnvstuplearn2;
net2.trainParam.goal = 0.1;
net2.trainParam.epochs = 2000;
[net2,tr] = train(net2,P2,T);
%=========================================================================
% 3.Trénování sítě
%=========================================================================
P3 =nnvstuplearn3;
net2.trainParam.goal = 0.01;
net2.trainParam.epochs = 1000;
[net3,tr] = train(net2,P3,T);
%=========================================================================
% 4.Trénování sítě
%=========================================================================
P =nnvstuplearn;
net3.trainParam.goal = 0.01;
net3.trainParam.epochs = 700;
[net3,tr] = train(net3,P,T);
vystup2 = sim(net3,nnvstup);
%=========================================================================
% 5.Trénování sítě
%=========================================================================
P3 =nnvstuplearn2;
net3.trainParam.goal = 0.01;
net3.trainParam.epochs = 500;
[net3,tr] = train(net3,P3,T);
%=========================================================================
% 6.Trénování sítě
%=========================================================================
P3 =nnvstuplearn3;
net3.trainParam.goal = 0.001;
net3.trainParam.epochs = 1000;
[net3,tr] = train(net3,P3,T);
%simulace
vystup3 = sim(net3,nnvstup)

vystup=(vystup1+vystup2+vystup3+vystup3)/4;
%=========================================================================
% Přiřazení výsledku ke klasifikační větě
%=========================================================================
X=vystup

Y=max(X(:))
Zs=sort(X);
Z=Zs(length(Zs)-1,1)
ZZ=Zs(length(Zs)-2,1)

veta{1,1}=('Extrasystola,příznak angíny Pectoris');%a
veta{2,1}=('Počáteční fáze/možnost výskytu angíny Pectoris. Jinak OK.');%b
veta{3,1}=('Respirační arytmie');%c
veta{4,1}=('Jemnovlnná fibrilace síní');%d
veta{5,1}=('Fibrilace síní(i hrubovlnná)');%e
veta{6,1}=('Zadržený dech');%f
veta{7,1}=('Podezření na plicní embolii,tachykardie');%g
veta{8,1}=('Pomalé,nebo zadržené dýchání,tachykardie');%h
veta{9,1}=('Fibrilace síní ');%i
veta{10,1}=('Hypokalémie');%j
veta{11,1}=('Hyperkalémie');%l
veta{12,1}=('Perikarditis / Infarkt Myokardu');%m
veta{13,1}=('Perikarditis / Myokarditis');%o
veta{14,1}=('Deprese ST intervalu => Angína Pectoris.');%p
veta{15,1}=('Slabá arytmie , OK ');%q

for i=1:length(X(:))
    if Y==X(i)
        vybervety=i;
        spravnost=Y;
        vysledek{1,1}=(veta{vybervety,1});
        vysledek{2,1}=(['Věrohodnost tvrzení : ' , num2str(spravnost) , ' z < 0 , 1 >. ']);
    end
    if Z > 0.2
    if Z==X(i)

        vybervety2=(i);
        spravnost2=Z;
        vysledek{4,1}=(['Alternativa : ',(veta{vybervety2,1})]);
        vysledek{5,1}=(['Věrohodnost tvrzení : ' , num2str(spravnost2) , ' z < 0 , 1 >. ']);
```

```
    end
    end
    if ZZ > 0.1
    if ZZ==X(i)

        vybervety3=(i);
        spravnost3=ZZ;
        vysledek{7,1}=(['Nepravděpodobně jde o : ',(veta{vybervety3,1})]);
        vysledek{8,1}=(['Věrohodnost tvrzení : ' , num2str(spravnost3) , ' z < 0 , 1 >. ']);
    end
    end

end

nnet=strcat(vysledek);
```

12.3 Soubor Otevrit.m

```
function varargout = Otevrit(varargin)
clc;
% OTEVRIT M-file for Otevrit.fig
%      OTEVRIT, by itself, creates a new OTEVRIT or raises the existing
%      singleton*.
%
%      H = OTEVRIT returns the handle to a new OTEVRIT or the handle to
%      the existing singleton*.
%
%      OTEVRIT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in OTEVRIT.M with the given input arguments.
%
%      OTEVRIT('Property','Value',...) creates a new OTEVRIT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Otevrit_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Otevrit_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Otevrit

% Last Modified by GUIDE v2.5 03-Jun-2008 16:35:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Otevrit_OpeningFcn, ...
                   'gui_OutputFcn',  @Otevrit_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% --- Executes just before Otevrit is made visible.
function Otevrit_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Otevrit
handles.output = hObject;
handles.Name=0;
handles.fileput=0; % to check if there's any input
handles.fileput1=0; % to check if there's any input
handles.fileput2=0;
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Otevrit_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%--------------------------------------------------------------------------
%               Nazev souboru - zobrazení
%--------------------------------------------------------------------------
function DataFile_Callback(hObject, eventdata, handles)

a=get(hObject,'string');
set(handles.DataFile,'string',a);
handles.fileput=1;
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function DataFile_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%--------------------------------------------------------------------------
%                Výběr souboru z adresářů
%--------------------------------------------------------------------------

% --- Executes on button press in FileBrowse.
function FileBrowse_Callback(hObject, eventdata, handles)
% hObject    handle to FileBrowse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[FileName,PathName] = uigetfile('*.txt','Select the TXT-file');
TotalFilename=fullfile(PathName,FileName);
set(handles.DataFile,'string',TotalFilename);
handles.fileput=1;
%handles.DataFile=handles.FileBrowse;
guidata(hObject,handles);

function FileBrowsemenu_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('*.txt','Select the TXT-file');
TotalFilename=fullfile(PathName,FileName);
set(handles.DataFile,'string',TotalFilename);
handles.fileput=1;
guidata(hObject,handles);

%--------------------------------------------------------------------------
%               Indikátor vytvořeného EKG souboru
%--------------------------------------------------------------------------

function indikekg_Callback(hObject, eventdata, handles)
%b=get(hObject,'string');
set(handles.indikekg,'string',b);
handles.fileput1=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function indikekg_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--------------------------------------------------------------------------
%                    Tlačítko   Make EKG File
%--------------------------------------------------------------------------

% --- Executes on button press in RUN.
function RUN_Callback(hObject, eventdata, handles)
% hObject    handle to RUN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

File_Name=get(handles.DataFile,'string');
filtrgraf=get(handles.exgraf1,'value')
specgraf=get(handles.exgraf2,'value')
%caszprac=get(handles.info2,'string')
caszprac=0;
[File_Name,caszprac,filtrgraf,specgraf] = EKG_open(File_Name,caszprac,filtrgraf,specgraf);
[pathstr, name, ext, versn] = fileparts(File_Name);
onlyname=joinseq(name,ext);
%fullna=joinseq(pathstr,'\')
%fullna=joinseq(fullna,onlyname);
set(handles.indikekg,'string',onlyname);
set(handles.DataFil,'string',File_Name);
set(handles.info2,'string',caszprac);
%caszprac=get(handles.info2,'string')
%msgbox('Nyní po stisku tlačítka RUN dojde ke klasifikaci.','Tip','help')
guidata(hObject,handles);

%--------------------------------------------------------------------------
%                 Tlačítko -  R U N
%--------------------------------------------------------------------------
% --- Executes on button press in spusti_QRS.
function spusti_QRS_Callback(hObject, eventdata, handles)
c=get(handles.DataFil,'string');
RRinterval=0;
probdet=0;
prenosna=0;
prenosna2=0;
caszprac=0;
```

```
timx=0;
Signal=0;
puk=0;
bod=0;
nnet=0;
[File_Name,prenosna,prenosna2,caszprac,timx,Signal,puk,bod,RRinterval,probdet,nnet]...
    = QRS_test(c,prenosna,prenosna2,caszprac,timx,Signal,puk,bod,RRinterval,probdet,nnet);
set(handles.info3,'string',prenosna);
set(handles.TS,'string',prenosna2);
set(handles.nnet,'string',nnet);
set(handles.info2,'string',caszprac);
%plot(timx(1:length(Signal)),Signal);
set(handles.Signal,'string',(Signal));
set(handles.timx,'string',timx);
set(handles.RRinterval,'string',RRinterval);
set(handles.problemy,'string',probdet);
set(handles.puk,'string',puk)


cla reset;
axes(handles.grafcelk);
cla reset;
b=timx(length(timx)-1)+2.5;
axis([0 b min(Signal) max(Signal)*1.2]);
plot(handles.grafcelk,timx(1:length(Signal)),Signal);
hold on;
    for bod=1:length(puk)
    plot(timx(puk(bod)),max(Signal(puk(bod)-5:puk(bod)+5)),'b.', 'MarkerSize',19);
    end


axes(handles.graflupa);
cla reset;
plot(handles.graflupa,timx(1:length(Signal)),Signal);
a=get(handles.posuvka,'Value')*timx(length(timx)-1)-0.2;
b=get(handles.posuvka,'Value')*timx(length(timx)-1)+2.5;
axis([a b min(Signal) max(Signal)*1.2]);
hold on;
for bod=1:length(puk)
plot(timx(puk(bod)),max(Signal(puk(bod)-5:puk(bod)+5)),'b.', 'MarkerSize',29);
tRvlna(bod)=timx(puk(bod));
end
%=========================================================================
        Plot_Title = strcat('EKG_', 2+47 , ' ze souboru : ', File_Name);
        Plot_Title = regexprep(Plot_Title, '_', '-');
        title(Plot_Title);
        xlabel('t [sec]');
        ylabel('U [mV]');
%=========================================================================
%-------------------------------------------------------------------------
%            Informační panel
%-------------------------------------------------------------------------

% --- Executes on selection change in info3.
function info3_Callback(hObject, eventdata, handles)
Signal=get(handles.Signal,'string');
timx=get(handles.timx,'string');
puk=get(handles.puk,'string');
Rvlnbod = (get(hObject,'Value'))-2
indik=str2double(puk(Rvlnbod,:))
    pomoccas=timx(indik,:)
    pomocsig=Signal(indik,:)
    pomocsig=max(str2num(Signal(indik-5:indik+5,:)))
axes(handles.graflupa)
plot(str2num(pomoccas),(pomocsig),'g.', 'MarkerSize',30);
    hold on;
axes(handles.grafcelk)
plot(str2num(pomoccas),(pomocsig),'g.', 'MarkerSize',20);
    hold on;
% --- Executes during object creation, after setting all properties.
function info3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%-------------------------------------------------------------------------
%            Informační panel - Čas běhu
%-------------------------------------------------------------------------
function info2_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function info2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




%--------------------------------------------------------------------------
%    Neviditelný objekt k indikaci QRSka
%--------------------------------------------------------------------------

function DataFil_Callback(hObject, eventdata, handles)

function DataFil_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Signal_Callback(hObject, eventdata, handles)

function Signal_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function timx_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function timx_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function puk_Callback(hObject, eventdata, handles)

function puk_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function info4_Callback(hObject, eventdata, handles)

function info4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%--------------------------------------------------------------------------
%     ---->>>   P O S U V N I K  --->>>
%--------------------------------------------------------------------------


% --- Executes on slider movement.
function posuvka_Callback(hObject, eventdata, handles)
%posuvhodnota = get(hObject,'Value')
Signal=str2num(get(handles.Signal,'string'));
timx=str2num(get(handles.timx,'string'));
posuv=(get(handles.posuvka,'Value'));
aa=(posuv*timx(length(timx)-1)-0.5)
bb=((posuv*timx(length(timx)-1))+1.5)
%aa=str2num(aa)
%bb=str2num(bb,10.0)
cc=min((Signal))
dd=max((Signal))*1.2
axes(handles.graflupa);
axis([aa  bb  cc  dd]);
%plot(handles.graflupa,str2num(timx(1:length(Signal))),str2num(Signal));

%hold on;
```

```
% --- Executes during object creation, after setting all properties.
function posuvka_CreateFcn(hObject, eventdata, handles)
% hObject    handle to posuvka (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

%--------------------------------------------------------------------------
%                 Š K R T Á T K A
%--------------------------------------------------------------------------

% --- Executes on button press in exgraf1.
function exgraf1_Callback(hObject, eventdata, handles)
filtrgraf=(get(handles.exgraf1,'Value'))


% --- Executes on button press in exgraf2.
function exgraf2_Callback(hObject, eventdata, handles)
specgraf=(get(handles.exgraf2,'Value'))


%--------------------------------------------------------------------------
%                 A B O U T
%--------------------------------------------------------------------------
% ---------------------------------------------------------------------
function Aboat_Callback(hObject, eventdata, handles)
zprava{1,1}=(' Název aplikace : Úprava a klasifikace EKG signálů');
zprava{2,1}=('                  Verze : 1.0');
zprava{3,1}=('Datum vytvoření : 25.5.2008');
zprava{4,1}=('                    Účel : Bakalářská práce');
zprava{6,1}=('                   Autor : David Loviška');
zprava{7,1}=('   Vedoucí práce : Ing.Jan Hrubeš');
msgbox(zprava,'Tip','warn')


%--------------------------------------------------------------------------
%                 H E L P
%--------------------------------------------------------------------------
% ---------------------------------------------------------------------
function navod_Callback(hObject, eventdata, handles)
zprava{1,1}=('Aplikace využívá externí výpočty.');
zprava{2,1}=('Proto je nutné postupovat následovně :');
zprava{4,1}=('1) Pomocí tlačítka Browse na horní liště,nebo uvnitř aplikace ');
zprava{5,1}=('    vybrat textový soubor se vstupními daty.');
zprava{6,1}=('    Musí se jednat o soubor *.TXT');
zprava{7,1}=('2) Nyní je nutné provést úpravy a filtrace vstupních dat.');
zprava{8,1}=('    Před tím lze zvolit,zda se má zobrazit průběh filtrovaného signálu a ');
zprava{9,1}=('    jeho spektrum včetně výsledného signálu.');
zprava{10,1}=('    To se provádí zaškrtnutím zatržítek.');
zprava{11,1}=('    Po nastavení provedeme úpravy stiskem tlačítka "Make EKG file".');
zprava{12,1}=('3) Samotnou klasifikaci provedeme tlačítkem "R U N".');
zprava{13,1}=('4) V okně "Výsledky" se objeví časy R-vln.');
zprava{14,1}=('    Poklepáním na čas R-vlny se nám daná vlna v grafu zvýrazní.');
zprava{15,1}=('5) Na spodním grafu je celý průběh a posunem posuvníku se nám ');
zprava{16,1}=('    v horním grafu zobrazí detail.');

msgbox(zprava,'Tip','help')


%--------------------------------------------------------------------------
%                 RR - interval - indikace
%--------------------------------------------------------------------------



% --- Executes on selection change in RRinterval.
function RRinterval_Callback(hObject, eventdata, handles)
Signal=get(handles.Signal,'string');
timx=get(handles.timx,'string');
puk=get(handles.puk,'string');
Rvlnbod = (get(hObject,'Value'))
indik=str2double(puk(Rvlnbod,:))
indik1=str2double(puk(Rvlnbod+1,:))
    pomoccas=timx(indik,:);
    pomoccas1=timx(indik1,:);
    pomocsig=max(str2num(Signal(indik-5:indik+5,:)))
    pomocsig1=max(str2num(Signal(indik1-5:indik1+5,:)))
```

```
axes(handles.graflupa)
plot(str2num(pomoccas),(pomocsig),'r.', 'MarkerSize',30);
plot(str2num(pomoccas1),(pomocsig1),'r.', 'MarkerSize',30);
    hold on;
axes(handles.grafcelk)
plot(str2num(pomoccas),(pomocsig),'r.', 'MarkerSize',20);
plot(str2num(pomoccas1),(pomocsig1),'r.', 'MarkerSize',20);
    hold on;


% --- Executes during object creation, after setting all properties.
function RRinterval_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%--------------------------------------------------------------------------
%                 Problemy - indikace
%--------------------------------------------------------------------------

% --- Executes on selection change in problemy.
function problemy_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function problemy_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




%--------------------------------------------------------------------------
%                 T a S vlny - indikace
%--------------------------------------------------------------------------


% --- Executes on selection change in TS.
function TS_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function TS_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%--------------------------------------------------------------------------
%                 Výstup umělé neuronové sítě
%--------------------------------------------------------------------------

% --- Executes on selection change in nnet.
function nnet_Callback(hObject, eventdata, handles)
% hObject    handle to nnet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns nnet contents as cell array
%        contents{get(hObject,'Value')} returns selected item from nnet


% --- Executes during object creation, after setting all properties.
function nnet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nnet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```